MSX

Graphic Workbook

グラフィック・ワークブック

ビジュアル体験で学ぶBASICプログラミング

桜田幸嗣・蓑島 聡 共著



アスキー出版局



Illustrate by Taki Ono
Design by Kenichi Samura／Number One Design Office

MSX

Graphic Workbook

グラフィック・ワークブック

ビジュアル体験で学ぶBASICプログラミング

桜田幸嗣・蓑島 聡 共著

アスキー出版局

本文デザイン……宇治　晶
本文イラスト……東恩納　裕一

イントロダクション

本書はFM-7グラフィック・ワークブックのMSX版です.グラフィックスを通して,MSXユーザーの方に中級程度のBASICプログラミングテクニックを身につけていただこうという目的で書かれました.

MSXのBASICに関しては,いままでにも『MSXビギナーズBASIC』を始め,数多くの入門書が出版されています.それらによって,BASICの基礎知識を得,次なるステップへいざ踏み出そうという方にとっては,本書のプログラムを読み,かつ理解することがプログラミングの良き練習となるでしょう.掲載されたグラフィックスの数々がその興味を一段と増すことは請け合います.

ところで,MSXという統一規格がパソコンのユーザーにとって大変なメリットを持つということは,すでにいろいろな場所で言及されています.

しかし,MSXの登場をこのうえもなく喜んでいる人々がもう1種類存在している,この事実は意外と知られておりません.そのもう1つの人種とは,何をかくそうパソコンの解説書執筆者たちであります.

すこし前までは,BASIC言語の説明をするにしても,グラフィックスを説くにしても,個々のメーカーの機種に特有な機能や制限のため,意を尽くした解説を行うことはなかなか難しかったものです.

パソコン一般について話を進めてしまうと,話題が画一的で,実際のマシン操作にはあまり役立たないものになりがちです.かといって,ある機種の特徴を十分に活かす説明を行おうと思うと,こんどは世界があまりに狭くなって,他に融通のきかない知識ばかり読者の方に与えているのではないか,という不安が生まれてきます.

その点MSXでは,このような1冊の書籍が,どのメーカーのどのMSXマシンを持っている方にも同じ条件の上に立ってお読みいただけるため,余計なことを考えずにこちらの伝えたいことをそのまま伝えられます.こうやって書き連ねたことが過不足なく読者の方々に応用していただけるわけで,筆者としてもこんなに喜ばしいことはありません.

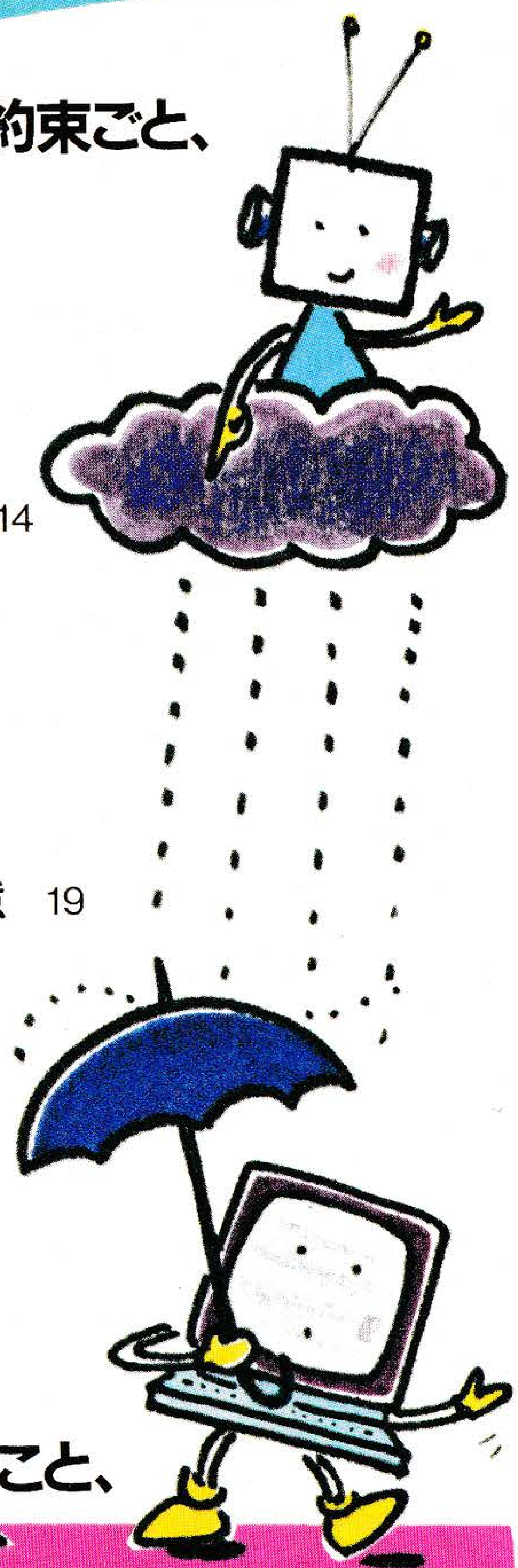
MSX関係の解説書を執筆している人間は,多かれ少なかれ同様な感慨を持っています.その意味で,MSXの本はどれをとってもその筆者なりの自信作となっているはず.本書もその例にもれません.我々は自信を持ってこの本をMSXユーザーの方々に捧げます.

昭和59年4月1日

桜田 幸嗣
蓑島　聡

MSXグラフィック・ワークブック

【目次】



1章　忘れていませんかグラフィックの約束ごと、MSXの使い方

グラフィックを始める前に　10

●画面モード　11
●画面モードとCOLOR命令　12

グラフィック命令に関する基本的御注意　14

●PSET, PRESETについて　15
●LINE or DRAW　15
●CIRCLEについて　16
●PAINT命令について　16
●色と画面モード　17

キャラクタ表示命令に関する基本的御注意　19

●WIDTHと趣味の画面モード　20
●グラフィックとキャラクタの混在　20
●グラフィックキャラクタについて　21

スプライト機能について　22

2章　グラフィックの基本操作……覚えてほしい、あんなこと、こんなこと、

位置を決める　30

●MSXで用いる座標　31
●中間点を求めて　34



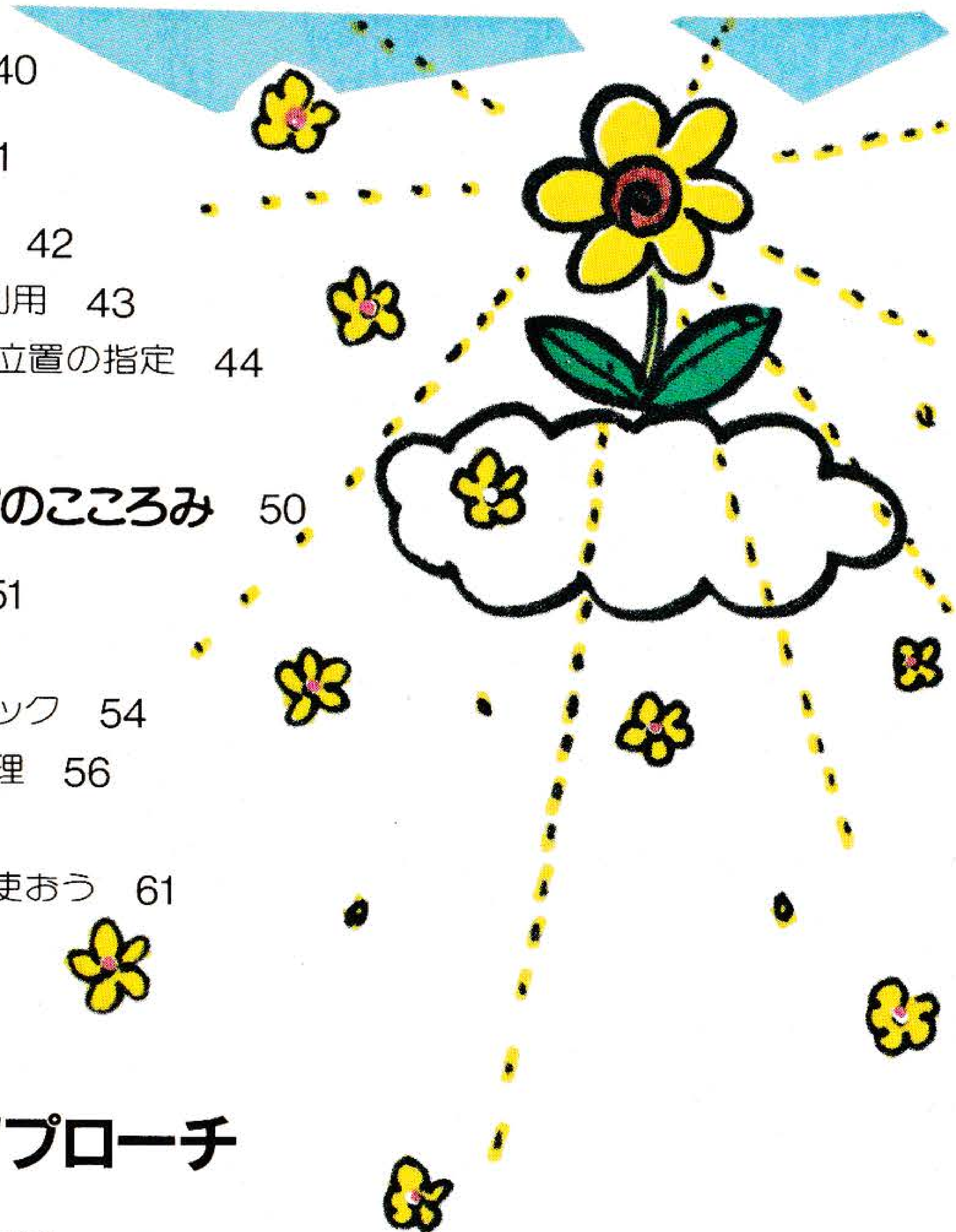
●図形の移動　37
●キャラクタ座標　40

「絵」をつくる　41

●コマンドを並べて　42
●READ DATAの利用　43
●角度と距離による位置の指定　44
●乱数の利用　47

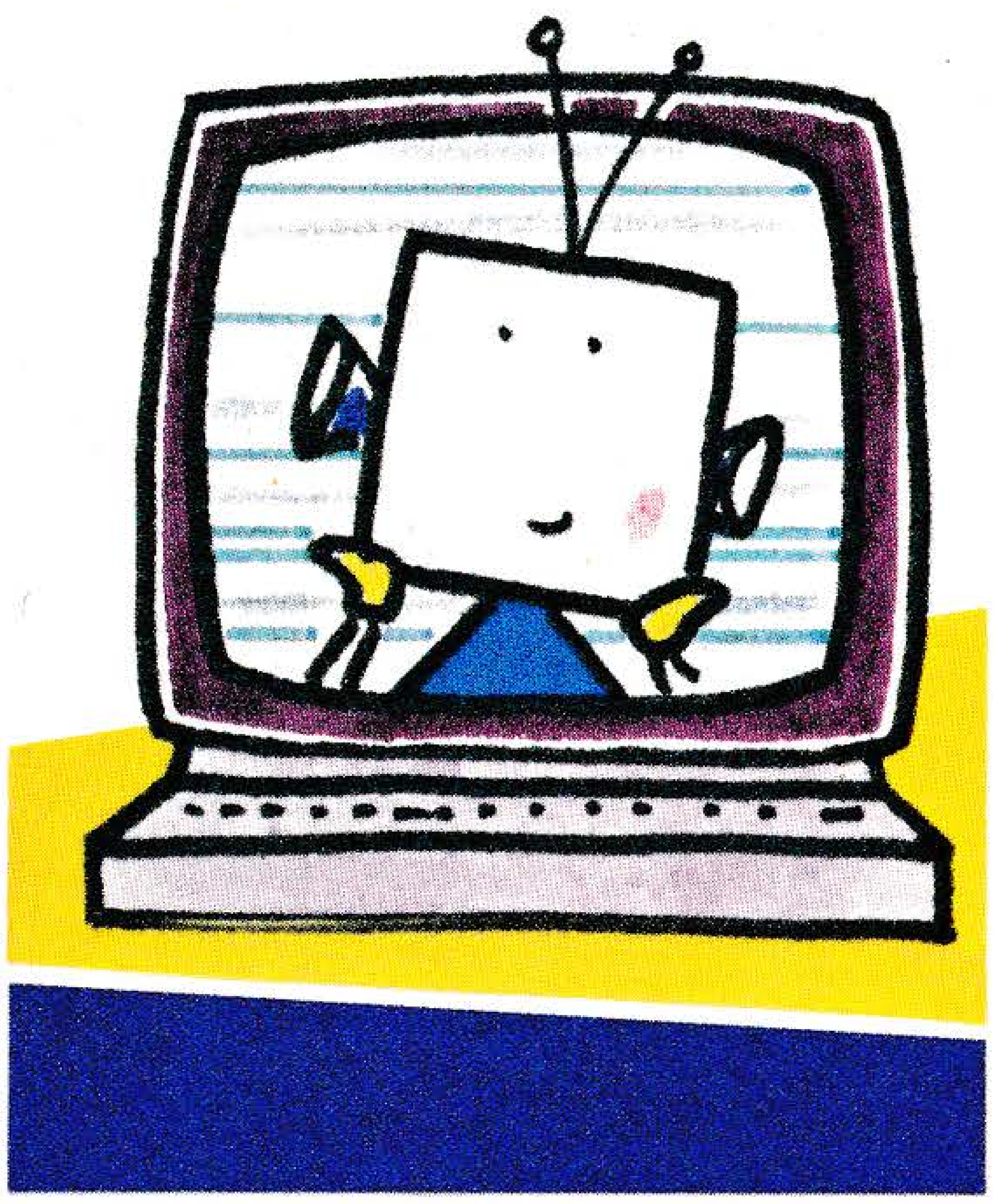
動くグラフィックのこころみ　50

●同じ形を並べて　51
●周期的繰返し　52
●移動するグラフィック　54
●キー入力による処理　56
●位置の判断　59
●スプライト機能を使おう　61



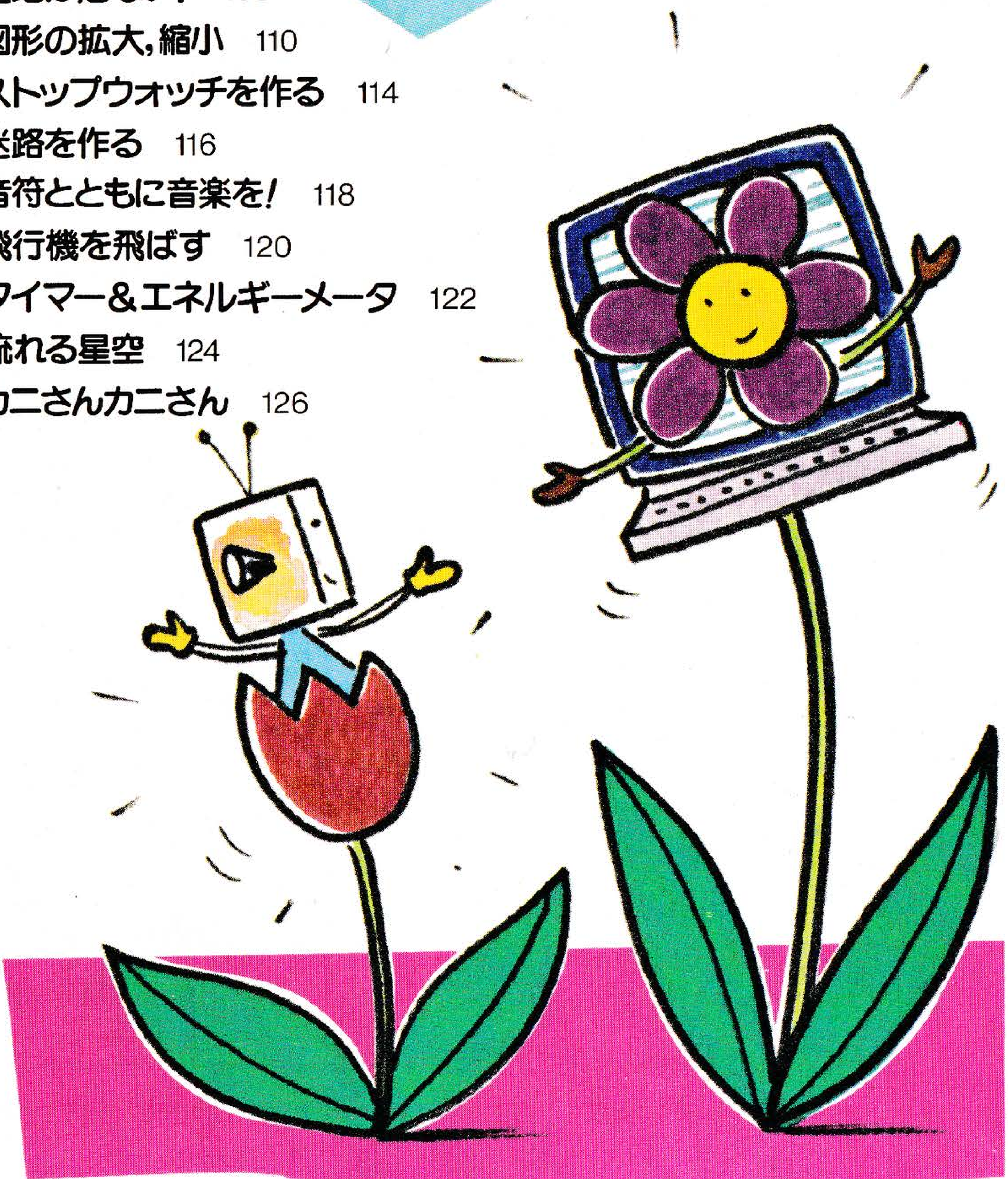
3章　アートへのアプローチ

直線で模様を描こう　68
MUG CUPS　70
スペース シャトル　72
リンゴを描く　74
落葉と星型　76
Stardust Night　78
円は異なもの味なもの　80
自分だけの曲線を描こう　84
ヘリコイド曲線を描く　86
ネコでも回そうか？　88
3次元グラフィックス？　90
文字の変形　94



4章　ゲームへのアプローチ

テニスゲームの仕組み　100
ハイスコアの表示　104
ビーム砲を打つ　106
基地が危ない!　108
図形の拡大,縮小　110
ストップウォッチを作る　114
迷路を作る　116
音符とともに音楽を!　118
飛行機を飛ばす　120
タイマー&エネルギーメータ　122
流れる星空　124
カニさんカニさん　126



5章　ランダムアプリケーション

バイオリズム　132
迷路ゲーム　137
焼けてないトタン屋根の上のネコ　143
スキーゲーム　153

Appendix

三次元グラフィック・パッケージ　164
スプライトパターン・ジェネレータ　171
MSX グラフィックエディタ　176



【1章】

忘れていませんかグラフィックの約束ごと、MSXの使い方

グラフィックを始める前に

- ●画面モード
- ●画面モードとCOLOR命令

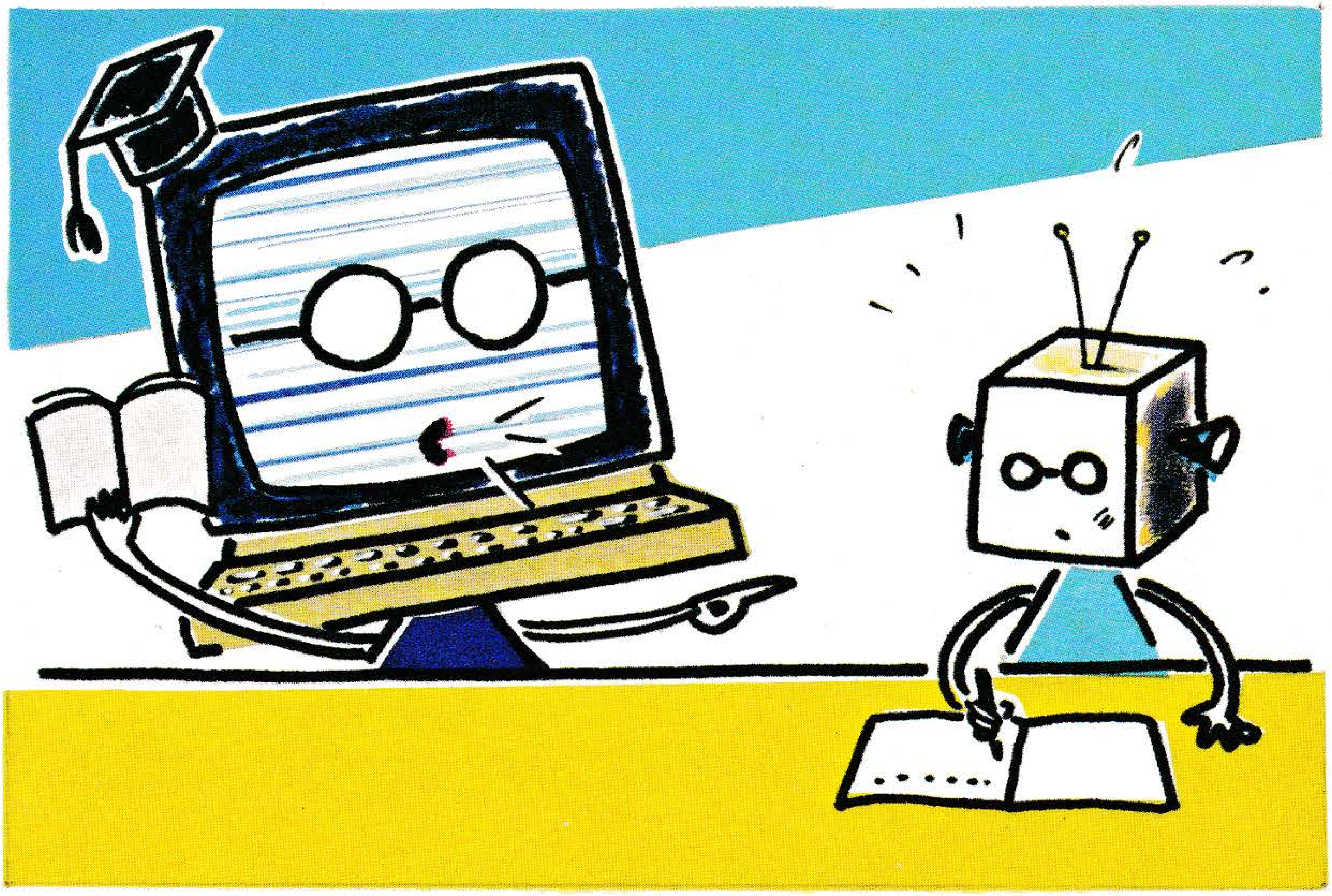
グラフィック命令に関する基本的御注意

- ●PSET, PRESETについて
- ●LINE or DRAW
- ●CIRCLEについて
- ●PAINT命令について
- ●色と画面モード

キャラクタ表示命令に関する基本的御注意

- ●WIDTHと趣味の画面モード
- ●グラフィックとキャラクタの混在
- ●グラフィックキャラクタについて

スプライト機能について



コンピュータ・グラフィックスの世界へようこそ．まずは手慣らし足慣らし，MSXのグラフィックス機能について，いろいろな話題を取り上げてお話しいたします．

実は企画段階では，本章はグラフィック手帳といいますか，個々のグラフィック命令の使い方を解説したマニュアル的雰囲気のページにしようと考えていたのです．けれども，原稿を書き進めていくうちに，筆者の心はむなしさで一杯になってきました．

こんなこと書かなくても，皆MSXのマニュアルに載ってることばかりではないか．同じことをここで書いたからって，読者の方には何も目新しいことはないんじゃないか．

そうなんです．だいたい，一冊の本の中でありとあらゆる説明を行おうとすることに，ムリ，ムラ，ムダがあるのです．ただでさえコンピュータ・グラフィックスには話のタネが多いもの．あれもこれもと欲張っていては，いつまでたっても本が終わりになりません．各命令の使用法に関しては，すでに立派なMSX-BASICリファレンスマニュアルというものが存在しています．これを見ていただけば十分ではありませんか．

というわけで，本章ではマニュアルをそのまま書き写したような説明は極力避け，それ以外の有益な小知識，あるいはコンピュータ・グラフィックスに関連したヨタ話などを，いくつか紹介いたします．どうぞ気軽にお読み流しください．ご参考となる部分が1か所でもあれば，満足であります．

グラフィックを始める前に

おや，おかしい．プログラムは絶対に正しく打ちこんだんだよ．ねー何度見たってマチガイないでしょ．なのになぜ写真と同じ絵が出てこないんだろう．うわぁーん．ボクのMSXが壊れちゃったよぉ．……などとお嘆きの貴兄に．あれ，そこのあなた，ヒトゴトのような顔しないでください．誰でもこういう事態におちいるかもしれないんですよっ．

というわけで，自分のMSXがどこかおかしいのでは，と心配しているあなた．いやいやパソコンなんてものは案外と丈夫にできてるものです．めったに壊れることなどありませんからご安心を．

正しいはずのプログラムが期待したように動かないときは，どこか画面表示に関する設定が変わっていると思って間違いありません．机に向かってMSXのスイッチを入れて，さぁこれからプログラムを作るぞぉー，という場合はよいのです．そんなときは本に載っているリストどおりに打ち込めば，まずトラブルは起こりません（もちろん正しく入力できていれば，の話ですが）．けれども，中には興じていたゲームが一段落して，いざ本でも読もうか，などと本書を手にする方もいるでしょう．予期せぬできごとが発生するのはそんなときです．

いわく，色がおかしい，文字が表示できない，などなど．それらは前に実行したプログラムによって，画面表示に関する設定が，いつもとは異なってしまったものがほとんどです．

このようなグラフィックの表示に関連したトラブルの原因とその解決法を，まとめてめんどうみようじゃないか，という次第で本節が設けられました．



Table 1

	文　字		グラフィック	
	英字，数字，カタカナ	グラフィック・キャラクタ ひらがな		
SCREEN 0	40×24文字	文字の一部がかける	表示不可	最大40×24文字 テキストモード
SCREEN 1	32×24文字	32×24文字	表示不可	最大32×24文字 テキストモード
SCREEN 2	OPEN "GRP：" FOR OUTPUT AS ＃n PRINT ＃n,〈文字列〉 CLOSE ＃n により表示可		256×192ドット	高解像 グラフィックモード
SCREEN 3			64×48ブロック	マルチカラーモード

＊1ブロックは4×4ドット

画面モード

MSXには，画面モードが4種類あります．この画面モードは，CRTもしくはTVに表示するのが文字なのか，グラフィックなのか，さらにひらがなは使うのか使わないのかなど，表示するものによって選ばなければなりません．

それぞれのモードは *Table 1* のような特徴があります．

MSXの電源をONにした状態では，SCREEN 1のモードで29×24文字表示となっています．これに対して，グラフィックを扱うときは，プログラム中でSCREEN 2または3を指定しなければなりません．なお，SCREEN 0または1では，いかなる方法でもLINE，PAINTなどグラフィック命令を実行することはできません．さらにグラフィック画面には，PRINT命令で文字を表示することはできません．しかし，後述の方法（1.3 グラフィックとキャラクタの混在，3.12 文字の変形）で，グラフィック画面に文字を表示することができます．

ここで，筆者の画面モード利用法を *Figure 1* に示しましたので参考にしてください．

最後にグラフィック画面は，プログラム中でSCREEN 2または3を実行することによって設定されますが，グラフィックを描き終った時点で自動的にテキスト画面に戻ってしまいます．これでは，せっかく描いたグラフィックも瞬時に消え去ってしまいますね．そこで本書では，END命令の前に

```
I$=INPUT$(1)
```

を挿入して，何らかのキーが押されるまでプログラムが終了しないようにしています．

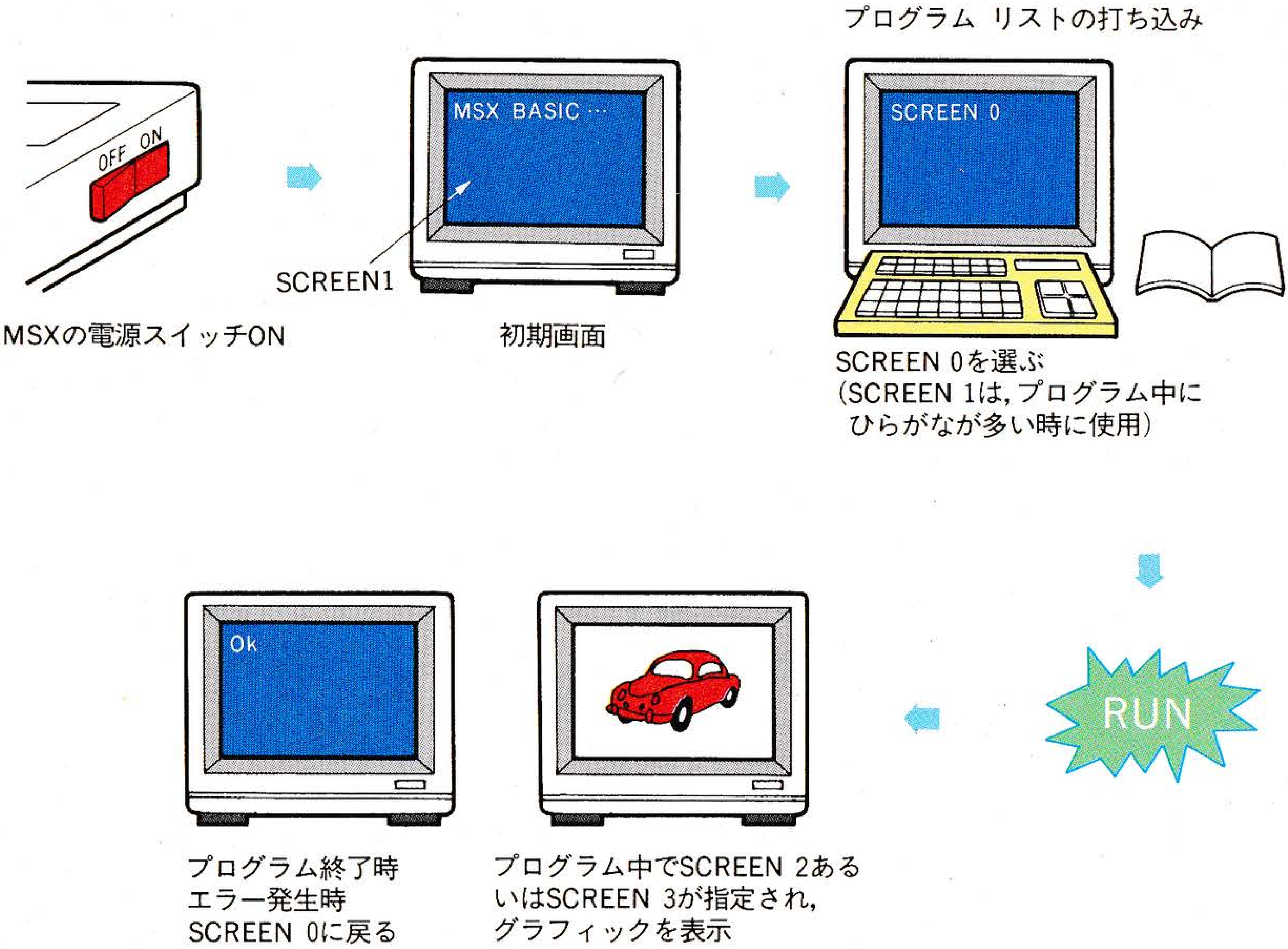


Figure 1

画面モードとCOLOR命令

MSXに画面モードが4種類あることは，もうすでにおわかりでしょう．この画面モードによってCOLOR命令は，指定方法が異なります（*Table 2*参照）．

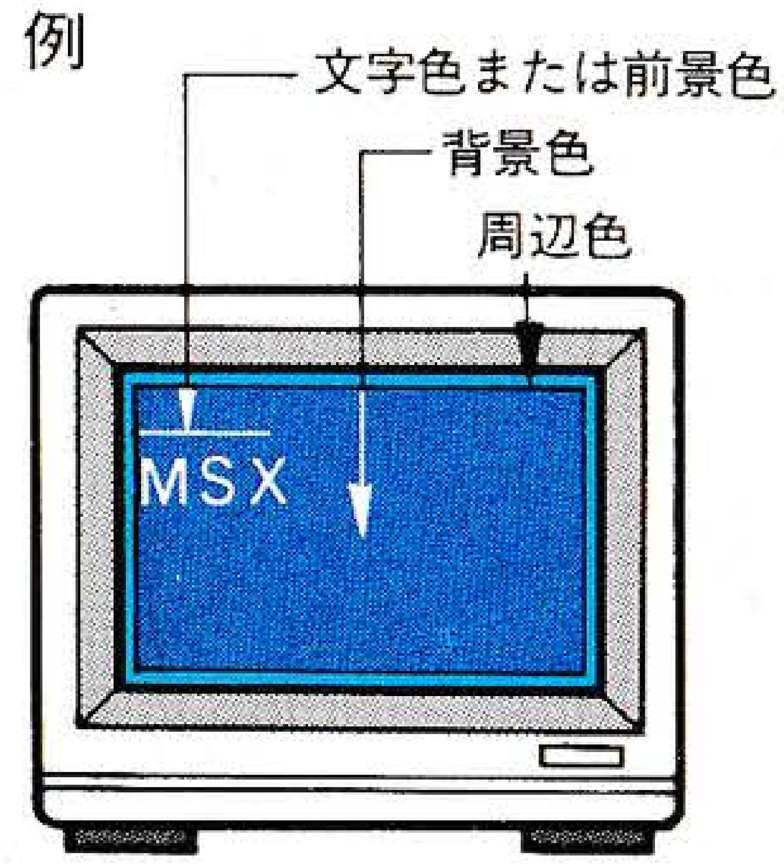
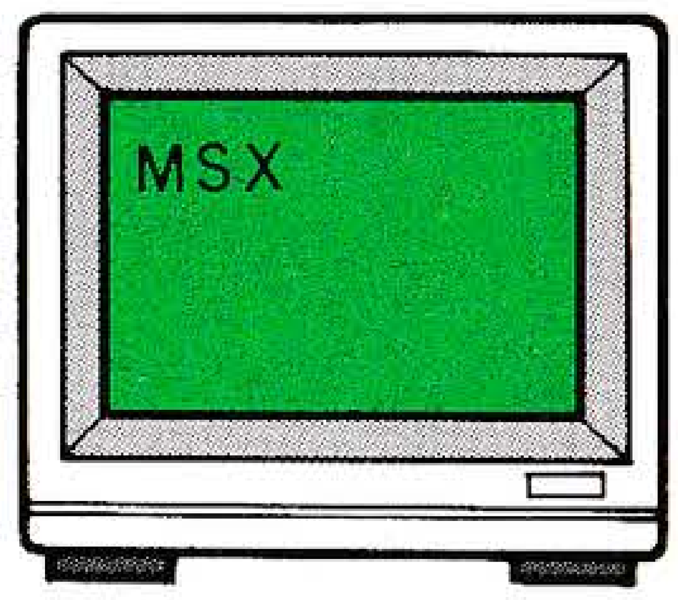
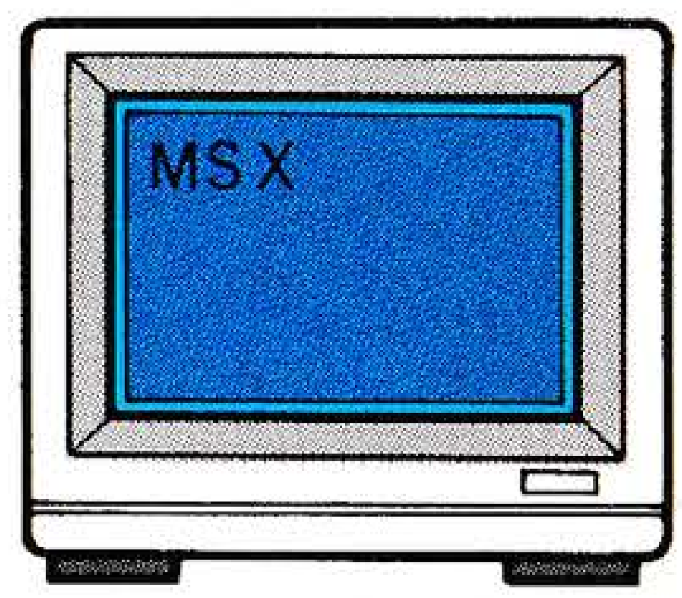
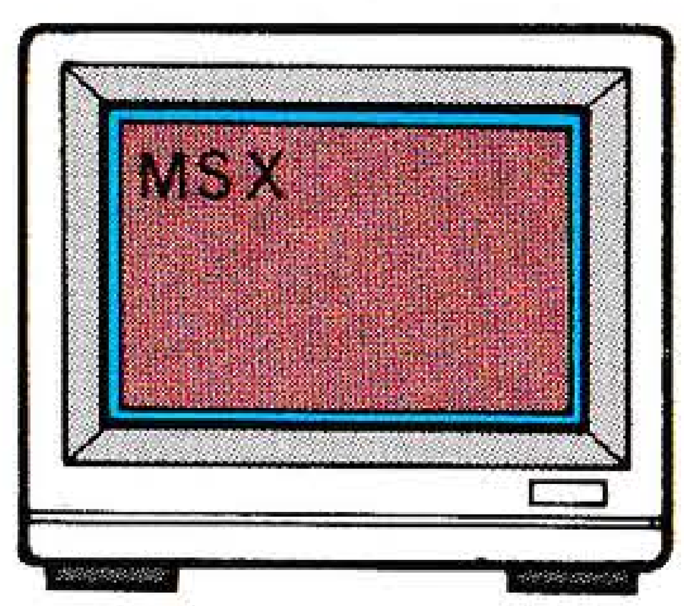
COLOR命令で実際に色が変わるのは，テキスト画面（SCREEN 0および1）では実行したときですが，グラフィック画面（SCREEN 2および3）では，次にCLS命令を実行したとき，または画面モードを変更したときです．

なお，MSXの電源をONにし，BASICでプログラミングができるようになったとき（BASIC起動時），画面モードは1，画面の色は文字が15(白)，背景が4(暗い青)，周辺が7（水色）に設定されています．COLOR命令を含んだプログラムを実行した後，プログラム・リストを見ようとしたが見えないもしくは見にくいときには，

```
SCREEN 1:COLOR 15,4,7:CLS
RETURN
```

をためしてみてください．

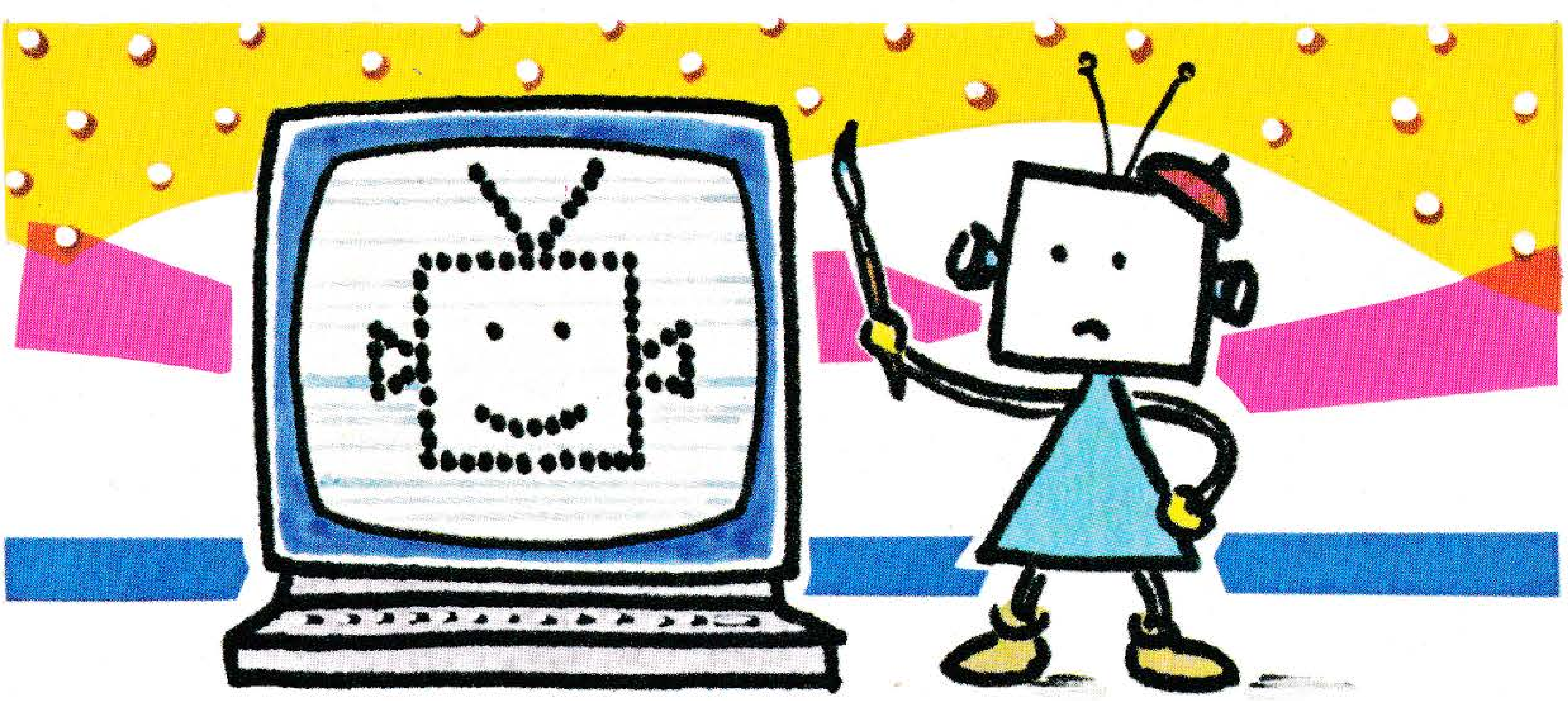
Table 2

画面モード	COLOR命令	例
		例 
SCREEN 0	COLOR a, b a：文字色 b：背景色	COLOR 1,2 
SCREEN 1	COLOR a, b, c a：文字色 b：背景色 c：周辺色	COLOR 1,4,5 
SCREEN 2 または, SCREEN 3	COLOR a, b, c a：LINE, CIRCLE命令などでカラーコード指定がない時，この色となる b：背景色 c：周辺色	COLOR 1,6,7 

グラフィック命令に関する基本的御注意

グラフィックの基本となる命令は，0次元の図形（"点"のことをもっともらしくいってるだけ）を描くPSETから，1次元の図形を描くLINEやDRAW，そして2次元の広がりを持つ図形を画面に表示し，かんたんに動かせるというSPRITE機能まで，各種とりそろえてあります．なんだ？3次元図形は描けないのか？——まあ今に立体テレビでも普及したら考えてみましょう．じゃ4次元図形は，ですって？あなた，この本はハインラインのSFではなくて，真面目なグラフィックの教科書なんですよ（いいすぎかな？）．

それはさておき，これらのグラフィック命令の使い方は，皆マニュアルに詳しく載っていますし，それをくどくどとここで蒸し返すのはやめておきましょう．ただ，これらの命令によって何ができるのかということ，また各命令を使う際に少しばかり注意しておいてもらいたいことなどを，いくつか取り上げて解説してみようというのが本節の目的です．こんなことわかってる，という方は読みとばしてください．



PSET, PRESETについて

PSET 命令は，指定した位置に指定した色で，点を1つだけ打つ命令です．MSX に限らず，パソコンのグラフィックはすべて点(ドットともいいます)を最小の単位として描かれます．ですからグラフィックを描くには，線を引くのも絵に色を塗るのも，原理的にいえば PSET 命令さえあれば何もいらないわけです．

しかし，それでは大きなグラフィック作品を描くことはまず不可能．そのため，線を引くには LINE，色を塗るには PAINT など，それぞれ便利な命令が用意されているのです．PSET 命令は，部分的にちょっと点でも打ってみようかという場合にのみ使いましょう（などと書いてしまうと，わたしは点描派グラフィック画家だとかいう人が現れて，PSET命令　だけで描いた「MSX 島の午後」なんていう大作を発表したりしてね．いえ，それはそれで賞賛いたしますよ）．

本書の中で使っている PSET 命令は，

> PSET (x座標, y座標), カラーコード

という書式で表します．もちろん色は，COLOR 命令で決めておき，PSET などの命令中では色指定を省略してしまう，という方法も可能なのですが，本書では個々のグラフィック命令の中で，色指定を行うという方針をとります．リストの打ち込みは多少めんどうになるかもしれませんが，この方が，確実にプログラムを理解していただけると思ったのです．

PSET が点を打つ命令であったのに対して，PRESET 命令は指定位置から点を取り除くという働きをします．その結果どうなるかといいますと，下地であるバックグラウンドカラーがその位置に現れることになります．

LINE or DRAW

2点間を結ぶ直線，あるいはその直線を対角線とする四角形を描く命令が LINE 命令です．その書式は次のようになります．

> LINE $(x_1, y_1)-(x_2, y_2)$, カラーコード, B指定

これに対して DRAW 命令は，いくつでも点をつないで線を延ばしていくことができるので，とても便利です．その書式は次のとおりです．

> DRAW<文字列>

この文字列は，グラフィックマクロ命令と呼ばれ，移動，回転，色指定，スケールファクタなどを決める文字，数字から構成されています．

これらの命令のうち，回転マクロ命令“A”とスケールファクタマクロ命令“S”が一度定義されると，次に定義しなおされるまでこの設定は変わりません．したがって，もし DRAW “S12” などを実行したのち，次にプログラムを LOAD し実行したとき，

DRAW "S4" がプログラム中にないと，原寸大で表示させたい図形が3倍もの大きさになって表示されてしまいます．こんなときは，

```
DRAW "S4A0"
```

を実行してください．グラフィックには関係ありませんが，これはPLAY命令のミュージックマクロ命令についてもあてはまり，音の大きさ，速さなどは，次に定義されるまでそのままなのです．

CIRCLEについて

円またはだ円，あるいはその一部分を描くCIRCLE命令は，グラフィックには欠かせないたいへんありがたい命令です．円を描くのは簡単なようにも思えますが，これが案外むずかしい作業でして，もしCIRCLE命令がなかったら，プログラムに要する労力は何倍にも増えてしまうでしょう．

CIRCLE命令の書式は，次のように，指定できるパラメータがとても多く，これらの使い方はなかなか覚えきれないかもしれません．そこで簡単に説明します．

```
CIRCLE (x,y),半径,カラーコード,
開始角度,終了角度,比率
```

(x，y) は中心の座標ですが，これを画面外に指定することはできません…当然．つまりMSXでは，xは0〜255，yは0〜191でなければならないのです．

半径は，x軸方向のドット数で指定します．しかし，だ円の場合，半径という値は，その長径を指定します．カラーコードは円の色を示します．

比率は，円のx軸方向の半径に対するy軸方向の半径の比で，大きくなるほど縦長のだ円が描けます．

開始角度と終了角度は，円の一周を0〜2πの範囲で，*Figure 2* のように表現した数値です．開始角度から終了角度に向けて反時計まわりに円が描かれます．

また，開始，終了角度に－(マイナス)を付ける（負の値にする）と扇型を描くことができます．－(マイナス)が付いていると円弧の両端と中心を直線で結びます．この－(マイナス)は，角度の値がマイナスではなく扇型を描くための印と考えてください．

PAINT命令について

このコマンドについては，SCREEN 2の高解像度グラフィックモードでは，境界色の指定ができず，領域色と同じ色で囲ま

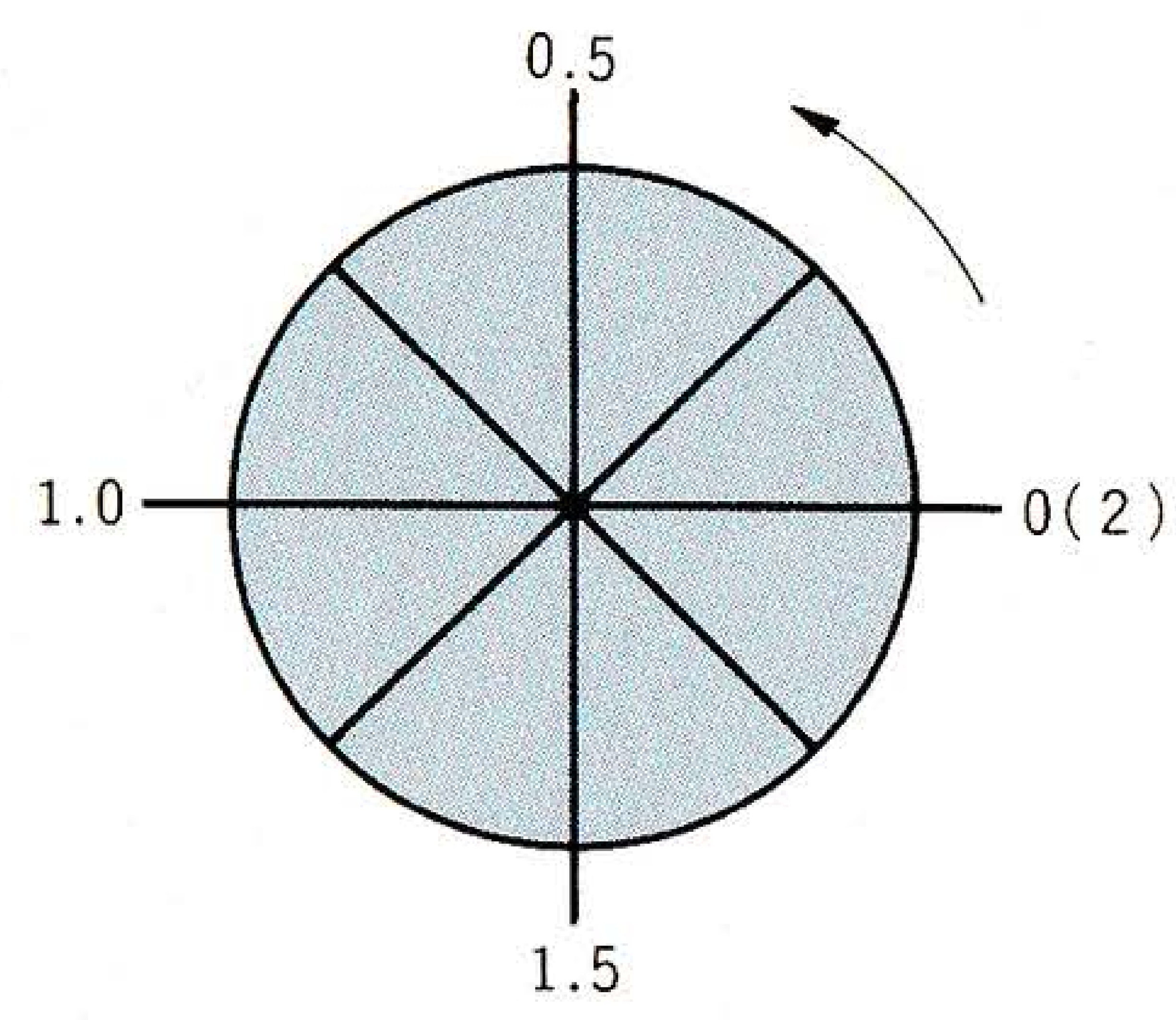


Figure 2

れた部分が塗りつぶされるという以外とりたてていうべきことも見つかりません．くわしくは，マニュアルを見てください．

色と画面モード

MSX では，16の色を使うことができます．この色は，*Table 3* のような0から15までの数（カラーコード）によって表されます．

SCREEN 3のマルチカラーモードでは，1ブロックごとにこの16色を自由に用いることができます．しかし，SCREEN 2の高解像度グラフィックモードでは，8ドットを1単位とする領域ごとに2色まで用いることができるという制約があります．この制約というのは，わかりにくいと思いますので，一例をあげて説明します．CRT 画面上で，y 座標が0，x座標が0〜255の横一列を考えます（*Figure 3*参照）．この列を8ドットずつ分けると0〜7，8〜15……と32分割されます．ここで，*Table 4* を見てください．ここで，COLOR ，13：CLS で背景色を紫にし，座標（1，0）に水色の点を打っておきます．次に座標（2 ，0）に同じ色の点を打った場合（a）は問題ありませんが，異なった色の点を打った場合（b）は座標（1，0）に打たれていたドットも変わってしまいます．そこで，分割された8ドット単位内が2色以上にならないようにグラフィックを描く必要があります．

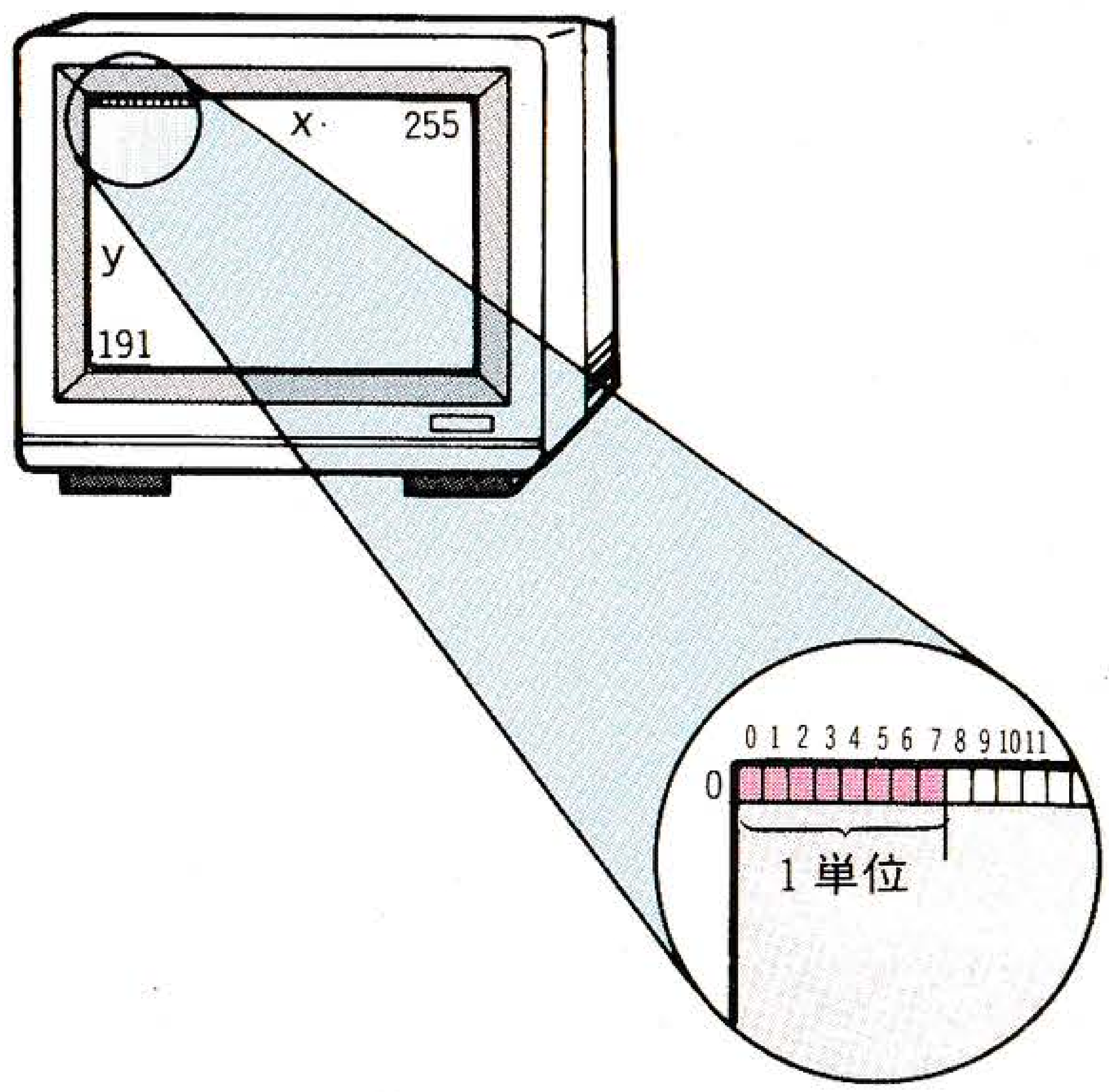


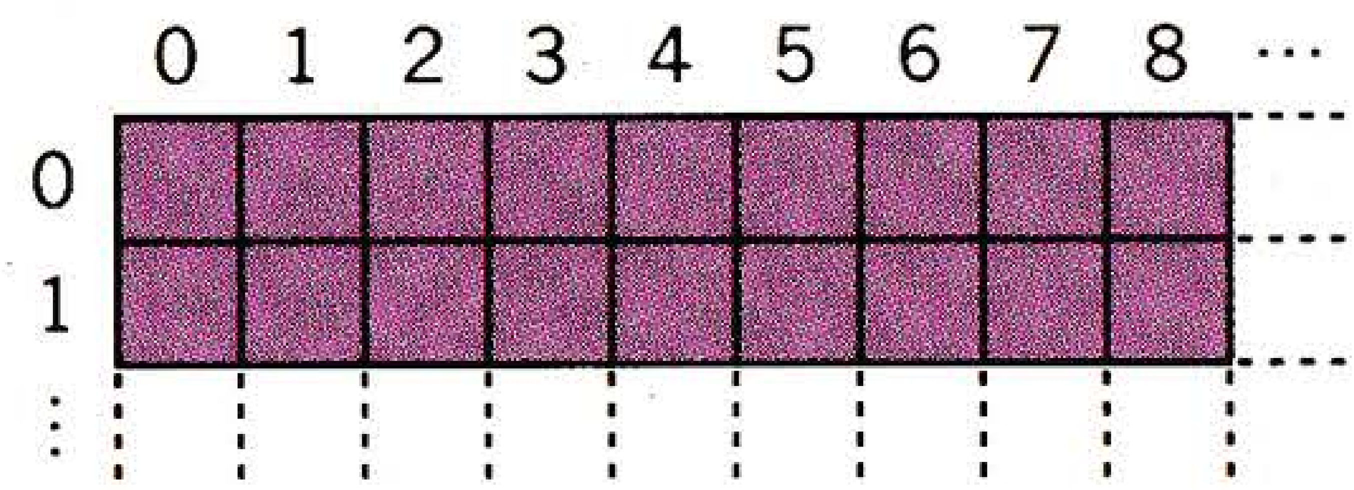
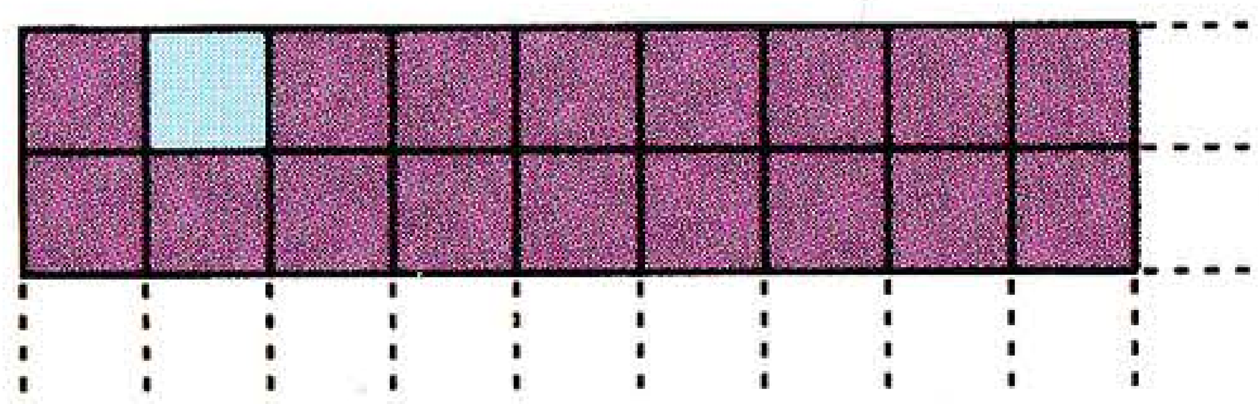
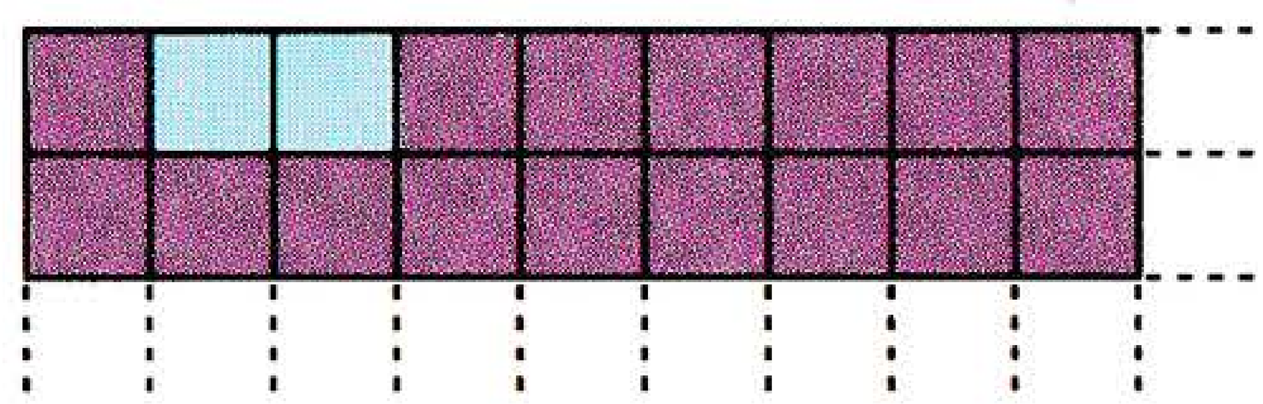
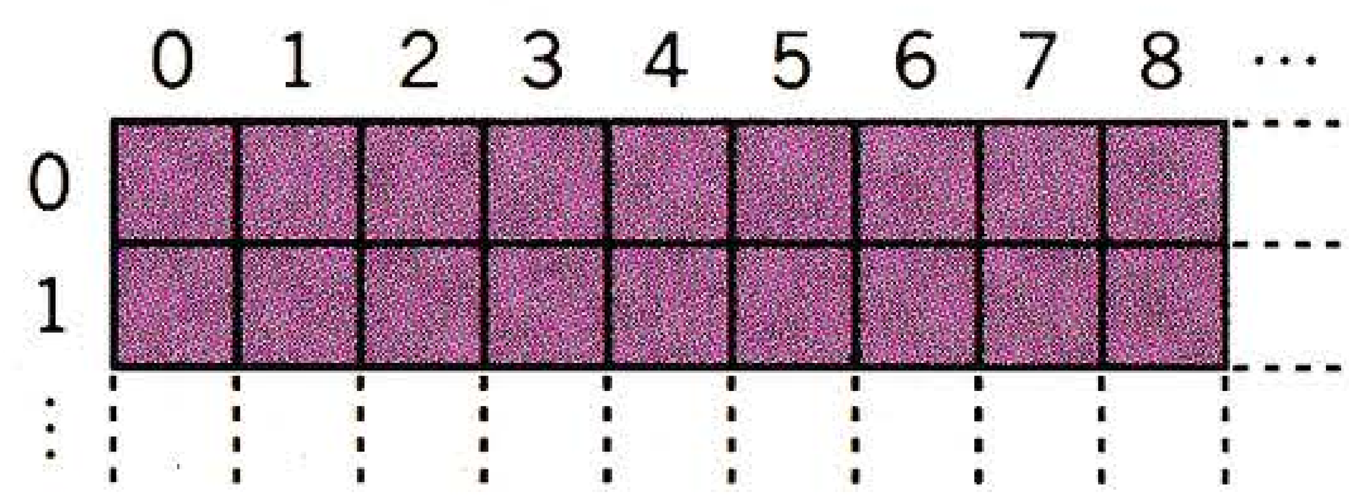
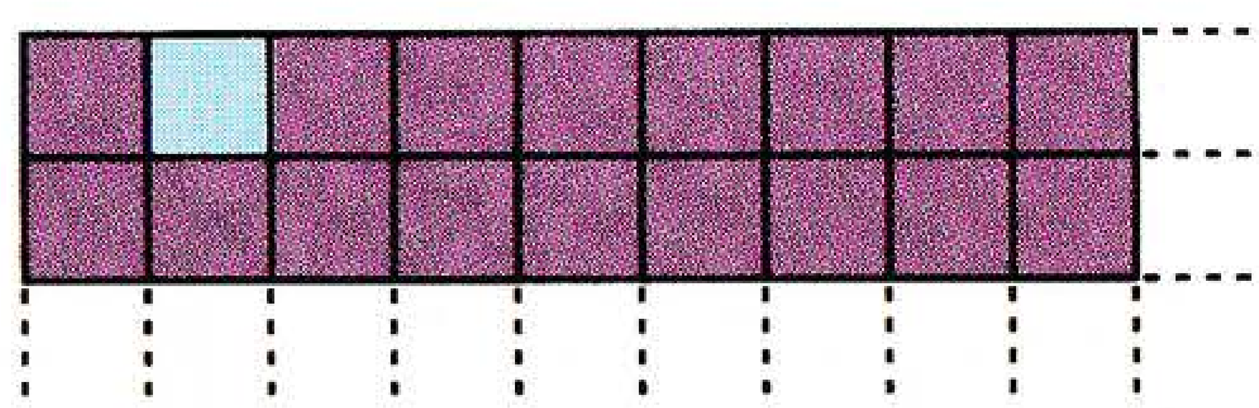
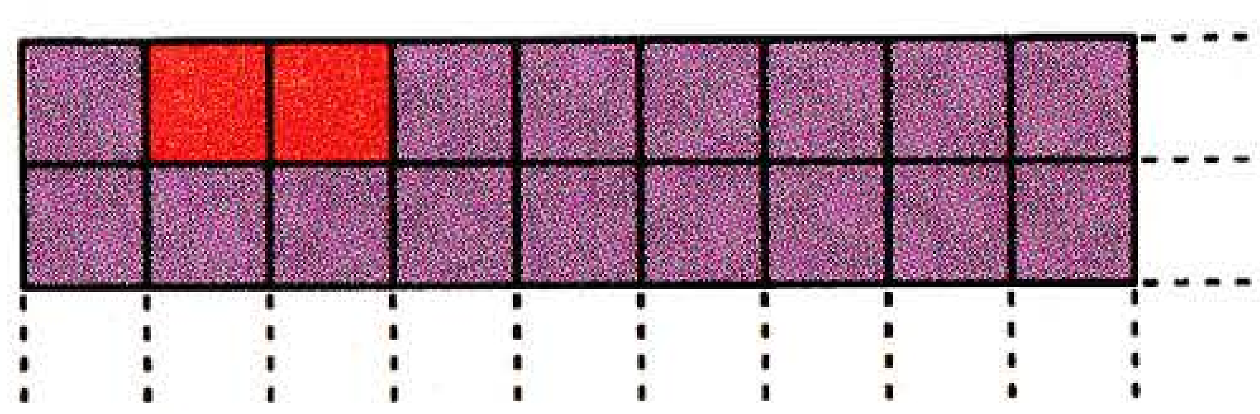
Figure 3

Table 3

カラーコード	0	1	2	3	4	5	6
色	透明	黒	緑	明るい緑	暗い青	明るい青	暗い赤

7	8	9	10	11	12	13	14	15
水色	赤	明るい赤	黄	明るい黄	暗い緑	紫	灰	白

Table 4

a	1		COLOR , 13 : CLS
	2		PSET (1 , 0) , 7
	3		PSET (2 , 0) , 7
b	1		COLOR , 13 : CLS
	2		PSET (1 , 0) , 7
	3		PSET (2 , 0) , 8

キャラクタ表示命令に関する基本的御注意

いったいグラフィックの本であるというのに，なぜ筆者はこんな文字表示についての項目の説明をしなきゃあならないんでしょう．グラフィックの話だけしてればいいじゃありませんか，読者の皆さんだってそれを望んでるはずですよね，ぶつぶつぶつ………．と，文句をいっていたら，編集者Y氏がこれを聞きつけました．

あんたはそういって，グラフィックにばかり肩入れしますがね，わたしゃ何の因果か本の編集なんて仕事をしているもんで，文字というメディアの重要性はよぉく知っているんだ．馬鹿にしちゃいけませんよ．人間が意志を伝え思想を伝え，文化を伝える一番の根っこなんですよ．文字ってのは．

ちょ，ちょっと待ってください．いえ何もそう大上段に振りかぶらなくたって．ただほんの少し，メンドクサイナ，なんて本当に頭の片スミでちらっと思っただけ．書きますよ．書けばいいんでしょう．そういえばエジプトの象形文字なんて，あれは大昔のグラフィックですよね．あはあは……．

あんたが作るグラフィックのプログラムにだって『リターンキーヲ　オシテクダサイ』なんて表示させる部分があるでしょ．もし文字が書けなかったらどうするの，「どうしてもリターンキーが押したくてたまらなくなるようなグラフィック」でも考え出してみますかね．

わかりましたよ，もう．コンピュータ・グラフィックスといえども文字の表示をまったく必要としないなんてことはない．だったらやはりそれを美しく画面に出す方法を説明しないのは，片手落ちではありますね．……と，納得した筆者は再びペンを取り，原稿用紙に向かうのでした．



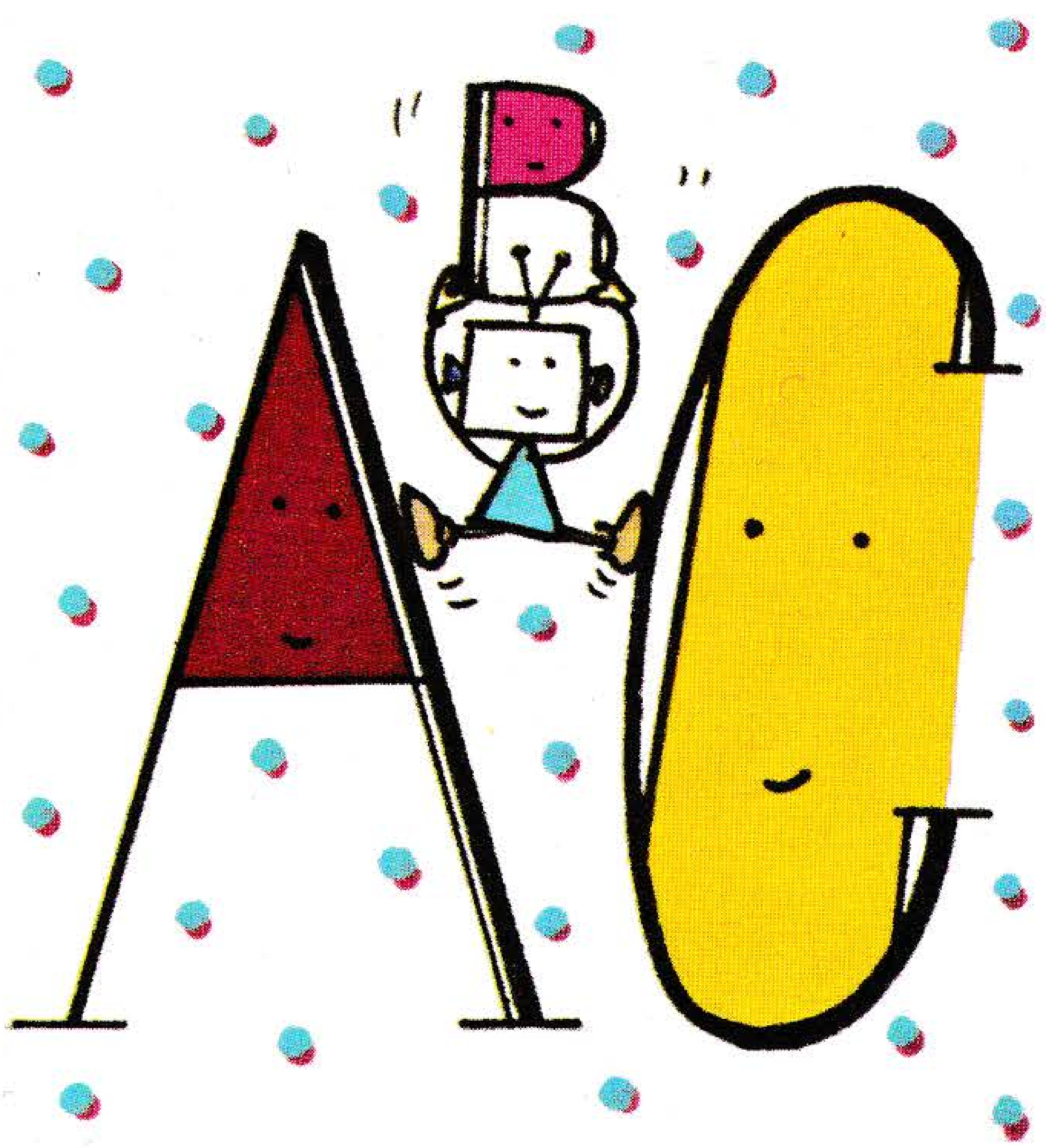
WIDTHと趣味の画面モード

画面の横1列に入る文字数を40字にするか32字にするか，それとももっと少なくするか，自分の趣味で選んでいただいてけっこうです．

なお，筆者らは，プログラムの打ち込みの際には

```
SCREEN 0:WIDTH 40 RETURN
```

により横一行に40文字入るモードを用いました．この40文字モードは，ひらがなの右端が欠ける，フォントが汚いなどの欠点もあります．しかし，プログラムを組む上では，1文字でも多く1画面内に入っていたほうが組みやすいので…要するに筆者らは，画面一杯に文字を詰めこむのが好きなのであります．そこで，このモードを利用しプログラムを作製したため，ひらがなよりカタカナを多く用いてしまいました．ひらがなの好きな方々，ぎっしりと詰めこまれた画面が見にくいという方々には，この場を借りて衷心よりおわび申し上げます．ドーモスイマセン．



グラフィックとキャラクタの混在

シューティングゲーム，ロールプレイングゲーム，シユミレーションゲーム，アドベンチャーゲーム……とゲームにはいろいろと種類がありますが，どのゲームにしてもスコアやその時々の情報を表示するためにキャラクタ（文字）を用います．ということは，グラフィックとキャラクタが一画面の中に混在しなくてはなりませんが，MSXの画面モードはテキスト用とグラフィック用に分かれています．しかし，MSXでは文字は，グラフィック画面でも扱うことができるのです．文字をグラフィック画面で扱うには，二三手続きが必要です．はじめに

```
OPEN "GRP:" FOR OUTPUT
AS #n································(a)
```

を実行し，これからグラフィック画面で文字を扱うということをコンピュータに宣言しておきます．このあとは，普通のPRINT命令と同様に，

```
PRINT #n·····························(b)
```

を使ってください．ただし，このときPRINT #n を使って表示させるべき文字は，LOCATE命令ではなく，DRAWなどによるグラフィック命令によって表示位置を定める必要があります．したがって文字の表示位置の指定もグラフィック座標による指定となるわけです．

表示し終わったら，コンピュータにグラフィック画面上で文字を取り扱うのを終るということを告げなければなりません．これが

```
CLOSE #n ·····························(c)
```

という命令です．なおEND命令がプログラムの最後にあれば，CLOSE #nは，プログラム中になくてもかまいません．

グラフィック画面上で文字を扱っているプログラムを実行している際，[CTRL]+[STOP]などで強制的にプログラムを終了させた場合，一応

```
CLOSE #n [RETURN]
```

を実行することをお進めいたします．

グラフィックキャラクタについて

グラフィックキャラクタをプログラム中で用いる際PRINT "月" などのように直接そのキャラクタを必要な命令中に書いてしまえば問題ありません．しかし，CHR$を用いてPRINT CHR$(n)とする場合*Table 5*に示すグラフィックキャラクタは，nがキャラクタコードのとき

```
CHR$(1)+CHR$(n+64)
```

としなければなりません．

CHR$ (1)〜CHR$ (31) は，キャラクタとコントロールコードの両方が割り当てられているため，頭にCHR$(1)が付いているか付いていないかによって，キャラクタなのかコントロールコードなのかをコンピュータが見わけています．そこでCHR$ (1) が付いている場合，後の文字はグラフィックキャラクタということになります．

Table 5

コード（10進）	キャラクタ	コード（10進）	キャラクタ
0		16	π
1	月	17	┴
2	火	18	┬
3	水	19	┤
4	木	20	├
5	金	21	┼
6	土	22	│
7	日	23	─
8	年	24	┌
9	円	25	┐
10	時	26	└
11	分	27	┘
12	秒	28	╳
13	百	29	大
14	千	30	中
15	万	31	小

スプライト機能について

D.C.2500，ここはベース・オブ・usr 0，第117ポート．係留中の愛機 vt100のコックピットで例のごとく居眠りをしていたパトロール隊中尉Nil Assoc（ニル アソック）は，突然鳴り響いたベルの音にたたき起こされた．

外敵侵入探知システムのアラームが点滅している．すぐさま発進，迎撃体勢を整えたのは，さすが筋金入りのパトロールである．しかし今日の敵はどうやらいつものやつらとは格が違うようだ．流星群も苦にせずヒョイヒョイ動き回るし，捕捉したかと思うとフッとワープして消えやがる．そうだ，まるで子供の頃ジイサンに聞いた小鬼の妖精のようだ．たしかスプライトとかいったっけ．こいつを捕えるのは，ちとホネだぞ……

おっと，そういうわけで唐突にSFまがいの書き出しですが，実はこれ，とある（架空の）世界におけるオープニング場面なのであります．ここに登場したパトロール隊員は，得体の知れない敵に少々てこずっている様子．障害物をものともせず，現れては消え，消えては現れる．この敵機の正体とは，さて何者なのでしょうか．

答はもちろんスプライト．MSXの偉大なる機能であるスプライトは，一口にいって次のような性質を備えています．

- 手頃な大きさ（8×8，あるいは16×16ドット）の図形である．
- 表示，消去，移動がとても簡単に行える．
- 表示し，移動させても背景の絵を消すことがない．
- 形は自由に定義できる．



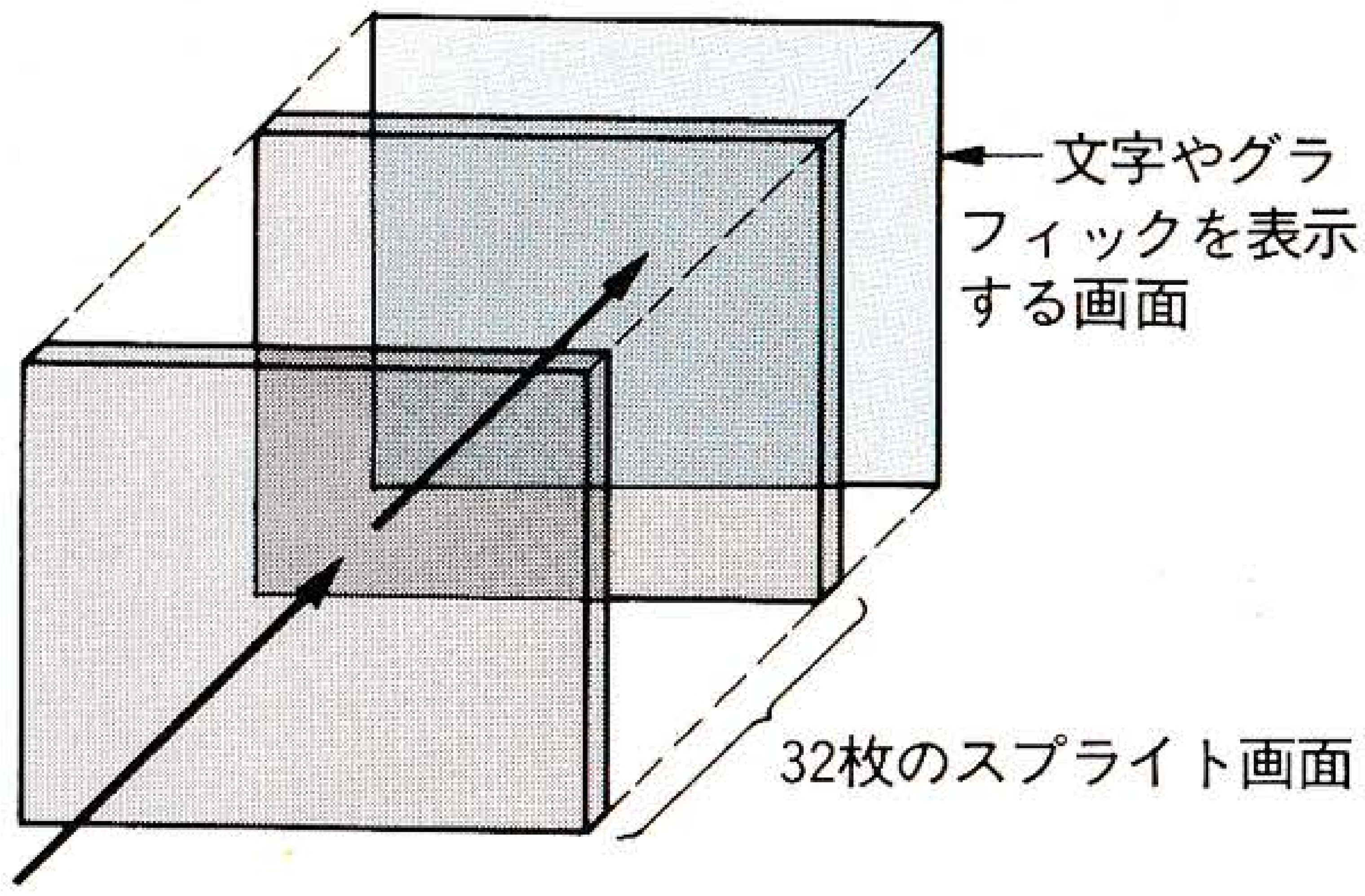


Figure 4

どうですか，ゲームの製作にはもってこいの機能ではありませんか．

この便利なスプライトについて知ってもらおうというのが本節の目的です．

ただし，本節ではその構成と機能の概略のみ掌握していただくことにして，実際のプログラム中での扱いなどは説明しません．そのあたりの実際的な使用法については，2章以降で取り上げることにいたします．乞，御期待．

さてまず，MSXには2種類の画面が存在する，ということから説明を始めましょう．その一つは文字やグラフィックを表示するための画面で，いわばメインスクリーンと呼ぶべきものです．この画面は，文字を表示したり，グラフィックで絵を描いたり，いろいろな用途に使うため，モードが4つあります．このモードを切り換えるための命令が SCREEN命令 です．

次にスプライト画面ですが，これはいわばメインスクリーン上に重ねられたガラス板でできたサブスクリーンと考えられるもので，32画面存在します．このスプライト画面上に表示された図形(スプライト図形)は，メインスクリーン（文字やグラフィック）の上に重ねて出力され，そのメインスクリーンを覆い隠しますが，メインスクリーン上の文字やグラフィックには何の影響も与えません(*Figure 4* 参照)．そこで，スプライト図形を取り去れば，再び隠れていたメインスクリーン上の文字やグラフィックが現れることになります（*Figure 5* 参照）．

またスプライト画面は，32面あり，CRTに向かって手前から順に番号（0〜31）が付いています．そして1スプライト画面に1個ずつ，計32個のスプライト図形を表示することができます．スプライト図形が重なった場合スプライト画面番号の小さいものほど手前にみえます（*Figure 6* 参照）．

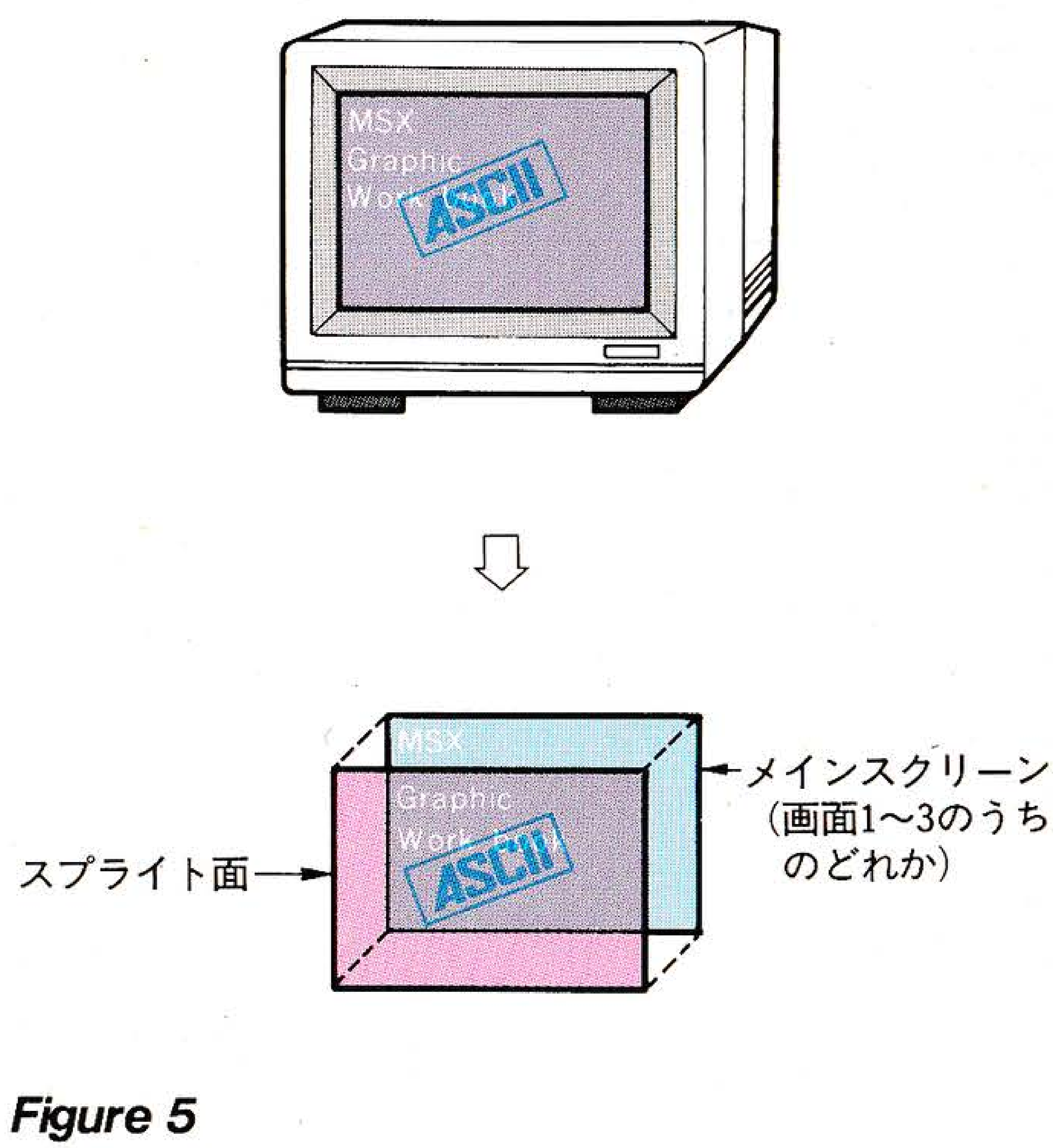


Figure 5

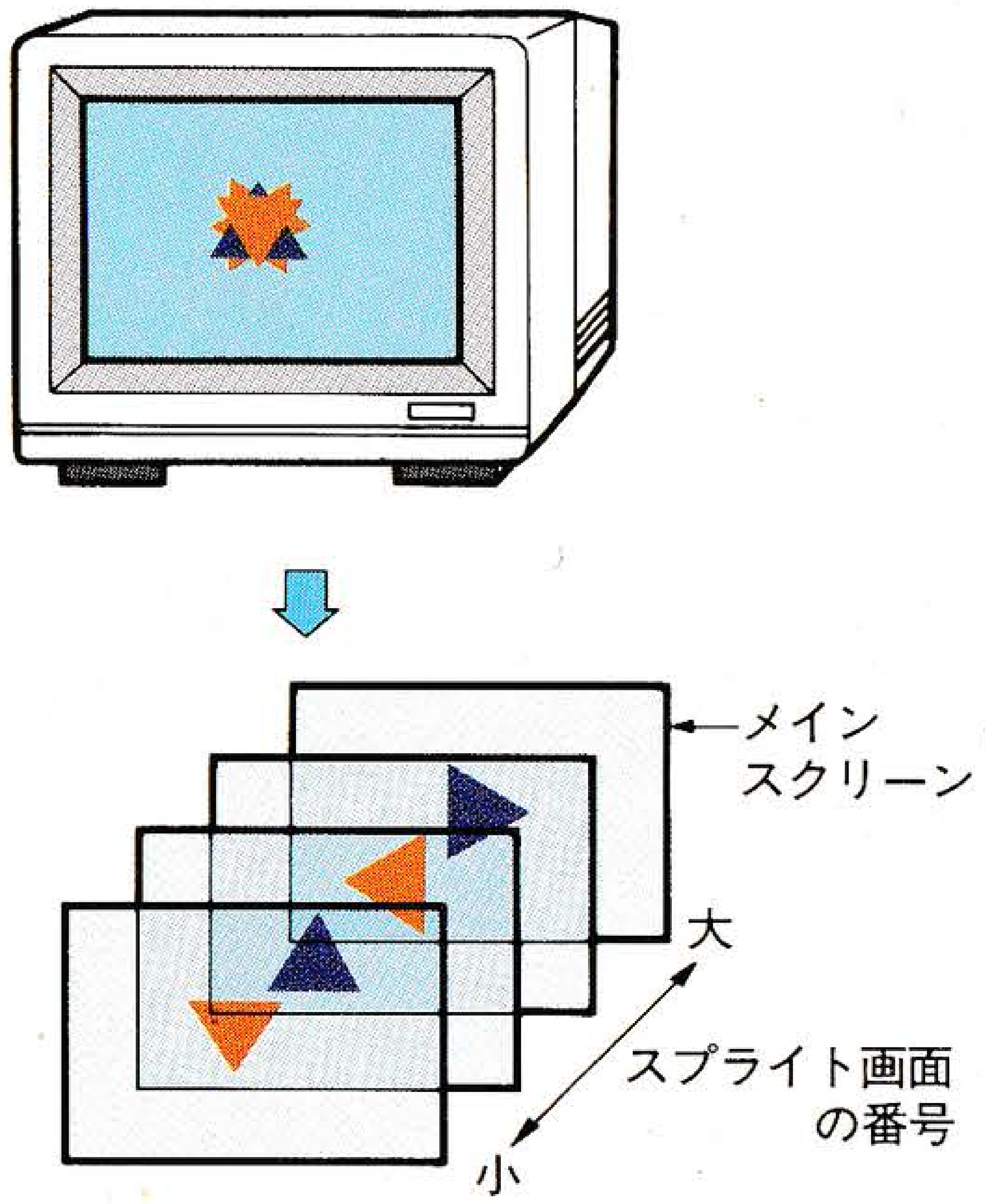


Figure 6

スプライト画面に表示するスプライト図形は，プログラムで自由に定義できます．定義された形はSPRITE$（n）という文字変数に蓄えられます．nはスプライト図形の番号で，以降この番号が定義した図形を表します．

たとえば，0番スプライト図形に星の形を，1番スプライト図形に月の形を定義しておいたとします．このとき，月とそれから10個の星を表示しようと思ったなら，0番スプライト画面（いちばん手前の位置）に1番図形（月）を置き，その後ろの1番から10番スプライト画面にそれぞれ0番図形（星）を置けばいいでしょう（*Figure 7* 参照）．

どのスプライト画面に何番のスプライト図形を置かねばならない，といった制限はありませんが，1つのスプライト画面に置けるスプライト図形はただ1個のみです．すでに図形が置かれているスプライト画面に，さらにもう1個のスプライト図形を置こうとすると，前にあったほうは外に追いやられて消えてしまいます．

スプライト図形の大きさは，8×8のサイズと16×16のサイズのどちらかのモードを設定できます．ただし，8×8を選んだ場合にはどのスプライト図形も8×8サイズでしか定義できませんし，16×16の場合も同じく16×16のスプライト図形しか使えません．

また，スプライト図形番号は，スプライト図形サイズによって，その上限がかわってきます．8×8のサイズでは0～255番まで，16×16のサイズでは0～63番までです．もっともスプライトパターンを64個も定義することはないでしょうから，あまり気にする必要はないとは思いますが．

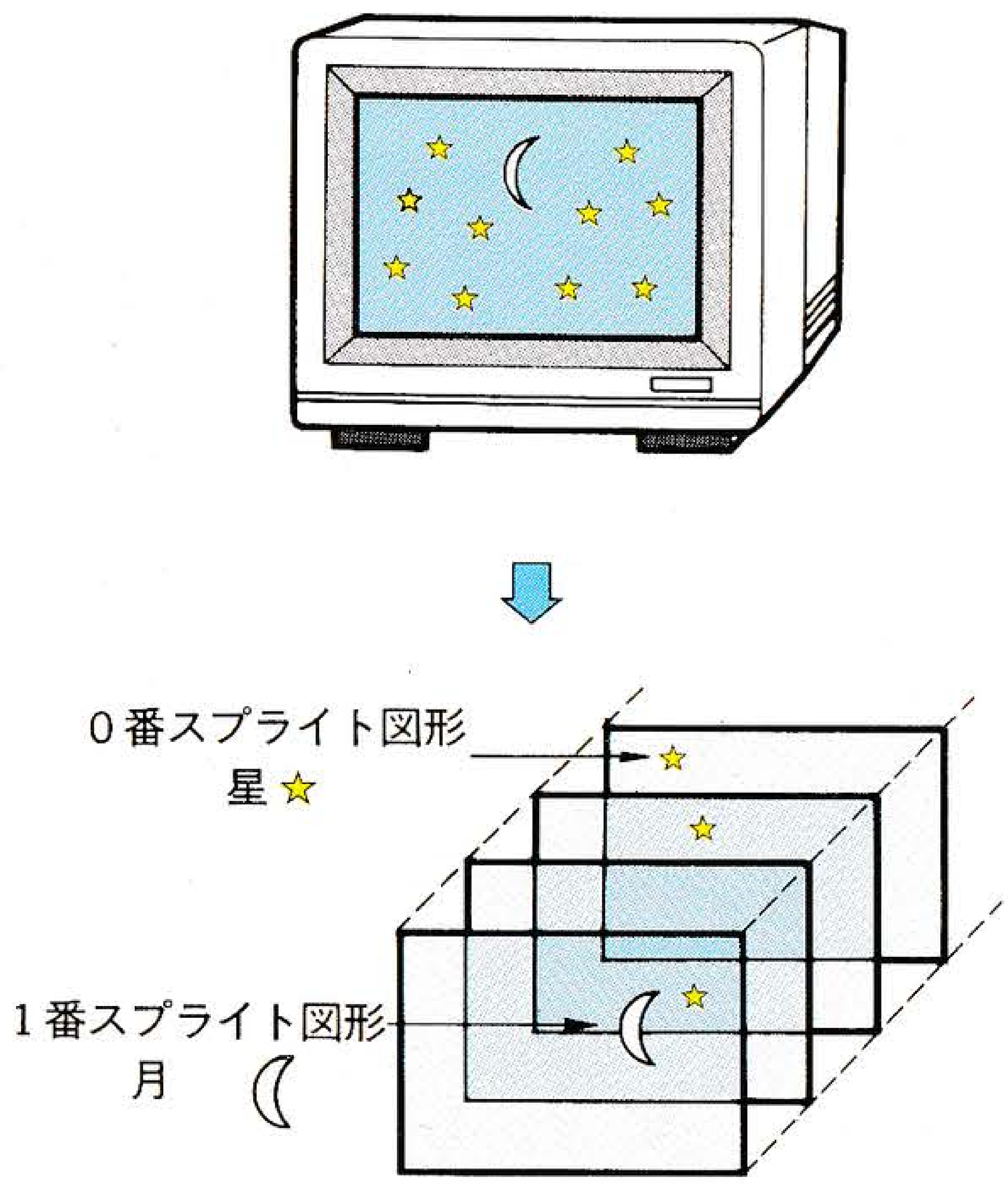


Figure 7

オタマジャクシを蛙に変える

学校で生物学の講座を履修すると，かならず『ウニの発生』などという映画を見せられます．ご存じない向きも多いでしょうが，あれは本当に感動的でして，機会があればぜひ一度は見ていただきたいと思います．

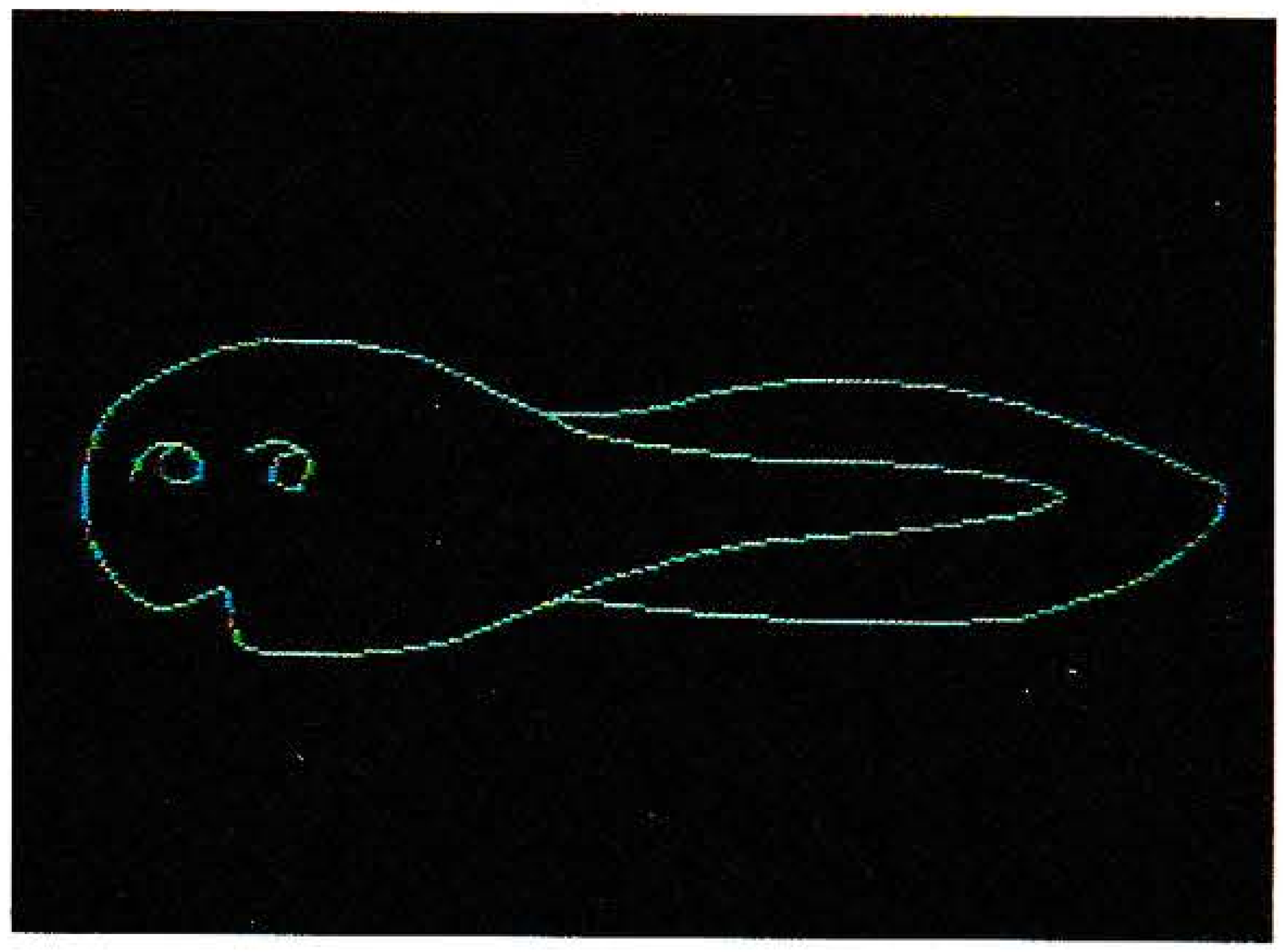
まず現れるのは多くの精子たち．そのなかの一匹がようよう卵子に到達すると，そのとたん透明なタンパク質の膜がふわっとひろがります．見ている人間たちも思わず息をのんでしまう第一の見せ場です．

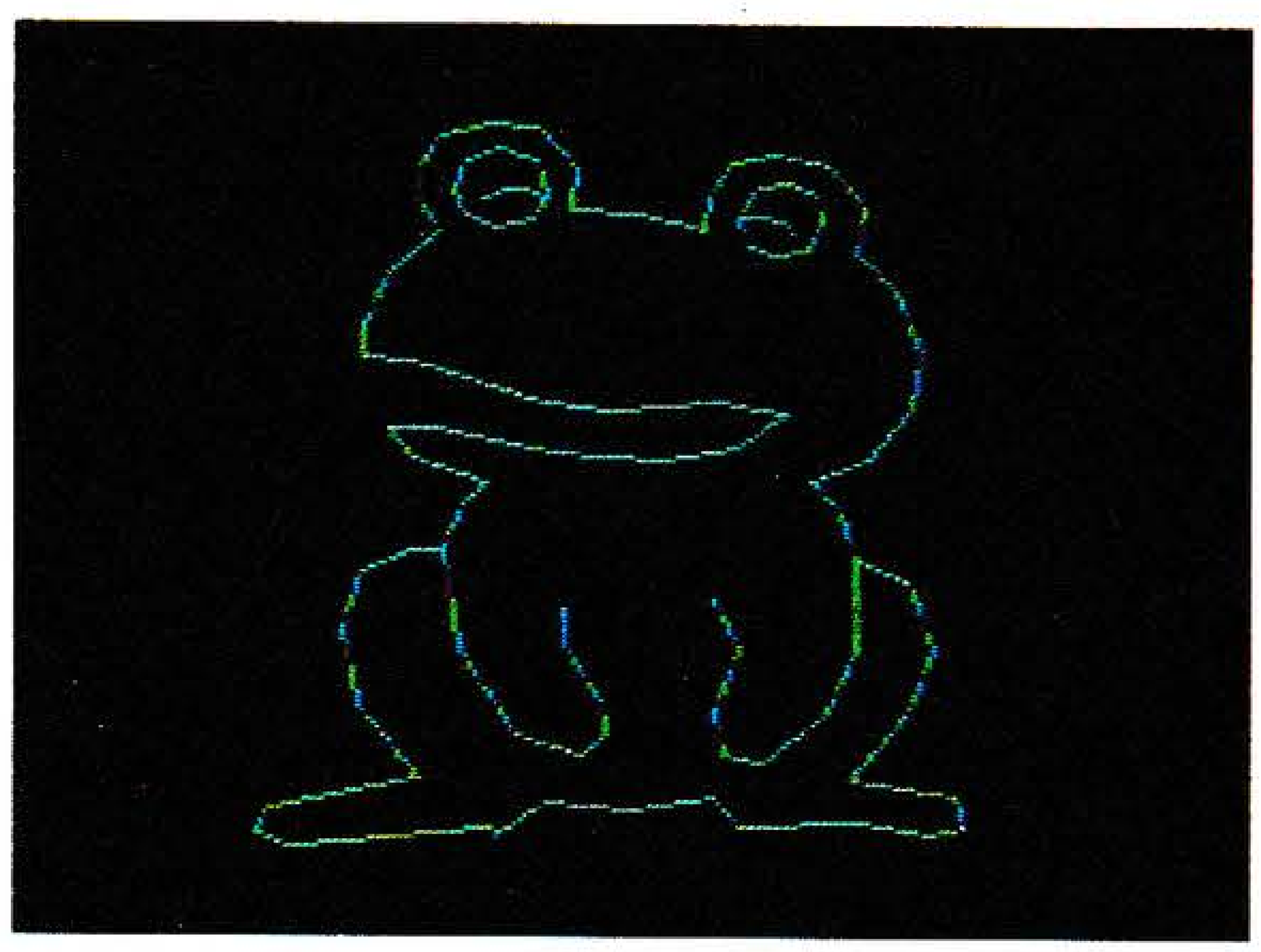
やがて卵子にくびれが生じ，卵割が始まります．だいたい，このようなフィルムは実際の何十倍かの速さで映写されるものですが，ハイスピードで見ていると卵子の分割する様子が，くりっくりっ，という感じで，これがナカナカかわいらしい．

ある程度卵割が進んで，個々の細胞の大きさが「目には定かに見えねども」の域に達するころになると，だんだん生き物らしい雰囲気がただよってきます．そうして，あちらがふくらみ，こちらが伸び，折れ曲がり，みるみるうちにウニの幼生の姿になっていくのです．

このような過程を目の前にみせられていると，なんと生命というものはうまくできているのかと，この世界を造りたまいし方の手際の良さには感心することしきりであります．

なんていう「まくら」を話の最初にふっておきながら，出てくるのがつまらぬプログラムで恐縮ですが，ここではオタマジャクシが蛙に変化する様子をグラフィックで





あらわしてみたいと思います．

といっても，なにも神の手になりかわって生物の発生をシュミレートしよう，などとだいそれたことを考えているわけではありません．

まず，出発点となる図形（オタマジャクシ）を用意します．またそれとは別に，最終的な図形（カエル）のデータも用意します．そして最初の図形からほんのすこしだけ最後の図形に近づいた形，それよりもさらにすこし最後の図形に近づいた形………を順に計算し，次々と描いていくとオタマ

ジャクシはいつしか立派なカエルに育つという次第.

両者の中間形を求めるため，次のような方法を用いることにしましょう．オタマジャクシもカエルもいくつかの点を線でつないで描いていくのですが，そのとき両者について点の個数をまったく同一にとり，それらの並ぶ順に1対1の対応をつけます.

ここで，それぞれの対応する点どうし，すなわちオタマジャクシを構成している点の第n番めのものと，カエルを構成している第n番めの点，これらの中点を計算してそれを順番に線で結んでいくと，そこに描かれるのは両者の中間形になるはずです.

さらに，両者のちょうど中点ではなく，もっとカエル側に寄った点を結んでいけばよりカエルに近い中間形が得られますし,オタマジャクシ側に寄った点を選べばそれだけオタマジャクシ的な中間形が得られます.

この考えでプログラムを作ると以下のリストのようになります．オタマジャクシからカエルへの華麗な変身（不気味な変身というウワサもちらほらありますが）をごらんください.

```
100 ' List Ex.1
110 '
120 DIM N(3),X0(3,49),Y0(3,49),X1(3,49),Y1(3,49)
130 SCREEN 2 : COLOR 3,1,1 : CLS
140 '
150 READ N(0),N(1),N(2),N(3)
160 FOR I=0 TO 3 : FOR J=0 TO N(I)
170   READ X0(I,J),Y0(I,J)
180 NEXT J,I
190 FOR I=0 TO 3 : FOR J=0 TO N(I)
200   READ X1(I,J),Y1(I,J)
210 NEXT J,I
220 '
230 FOR M=0 TO 1 STEP 1/8
240   CLS
250   FOR I=0 TO 3 : FOR J=0 TO N(I)
260     X=X0(I,J)*(1-M)+X1(I,J)*M
270     Y=Y0(I,J)*(1-M)+Y1(I,J)*M
280     IF J=0 THEN PSET (X,Y) ELSE LINE -(X,Y)
290   NEXT J,I
300   FOR T=1 TO 500 : NEXT T
310 NEXT M
320 I$=INPUT$(1):END
330 '
340 ' ----------------
350 '
360 DATA 30,46,11,11
370 '
```

```
380 ' ----------------
390 '
400 DATA 210,100,202,104,180,108,160,110,145,112,134,115,120,120
410 DATA 105,127,109,126,114,126,120,127,140,129,160,130,180,130
420 DATA 200,129,220,122,230,117,240,109,244,104,245,100,243, 96
430 DATA 231, 90,216, 84,200, 78,183, 74,168, 73,155, 73,139, 76
440 DATA 128, 80,117, 82,105, 82
450 '
460 DATA 210,100,205, 97,190, 94,169, 92,150, 92,125, 89,111, 86
470 DATA 105, 82, 99, 79, 93, 76, 88, 73, 79, 69, 74, 68, 69, 67
480 DATA  64, 66, 58, 66, 52, 66, 48, 66, 43, 67, 39, 68, 34, 70
490 DATA  28, 73, 22, 77, 16, 82, 13, 87, 11, 94, 10,100, 10,106
500 DATA  11,113, 14,120, 17,124, 20,127, 25,130, 27,130, 30,129
510 DATA  33,128, 36,126, 39,125, 40,127, 41,133, 42,137, 44,139
520 DATA  50,140, 65,140, 80,138, 90,135,105,127
530 '
540 DATA  29, 91, 26, 94, 26, 98, 28,100, 31,100, 34, 99, 34, 95
550 DATA  32, 92, 29, 91, 25, 91, 21, 95, 20,100
560 '
570 DATA  56, 92, 51, 93, 49, 96, 49, 99, 52,101, 55,101, 57, 97
580 DATA  57, 95, 56, 92, 52, 90, 48, 90, 44, 92
590 '
600 ' ----------------
610 '
620 DATA 115,130,115,139,122,152,123,160,118,165,105,160, 94,140
630 DATA  91,116, 85,116, 74,121, 70,135, 73,153, 85,170, 54,177
640 DATA  51,183, 58,186, 99,182,100,184,110,176,128,178,145,175
650 DATA 150,182,175,181,190,184,197,182,195,175,173,171,185,155
660 DATA 190,140,185,124,174,119
670 '
680 DATA 145,128,150,140,145,155,147,165,155,167,168,155,174,140
690 DATA 174,119,171,108,165,100,177, 95,186, 83,188, 70,183, 55
700 DATA 175, 46,177, 35,170, 25,159, 22,150, 24,145, 32,144, 40
710 DATA 131, 36,117, 35,118, 25,110, 15,100, 14, 90, 17, 86, 25
720 DATA  86, 33, 90, 40, 80, 50, 75, 63, 75, 70, 85, 71,112, 81
730 DATA 125, 83,146, 81,160, 84,150, 92,135, 95,113, 93, 90, 87
740 DATA  80, 87, 85, 94,100,100, 95,105, 91,116
750 '
760 DATA 112, 32,110, 23,102, 20, 95, 23, 93, 30, 95, 36,102, 39
770 DATA 109, 37,112, 32,108, 30,104, 30, 99, 32
780 '
790 DATA 151, 37,152, 43,157, 46,165, 45,168, 39,167, 32,162, 29
800 DATA 155, 30,151, 37,155, 37,160, 38,162, 41
```

【2章】

グラフィックの基本操作……
覚えてほしい、あんなこと、こんなこと、

位置を決める

- ●MSXで用いる座標
- ●中間点を求めて
- ●図形の移動
- ●キャラクタ座標

「絵」をつくる

- ●コマンドを並べて
- ●READ DATAの利用
- ●角度と距離による位置の指定
- ●乱数の利用

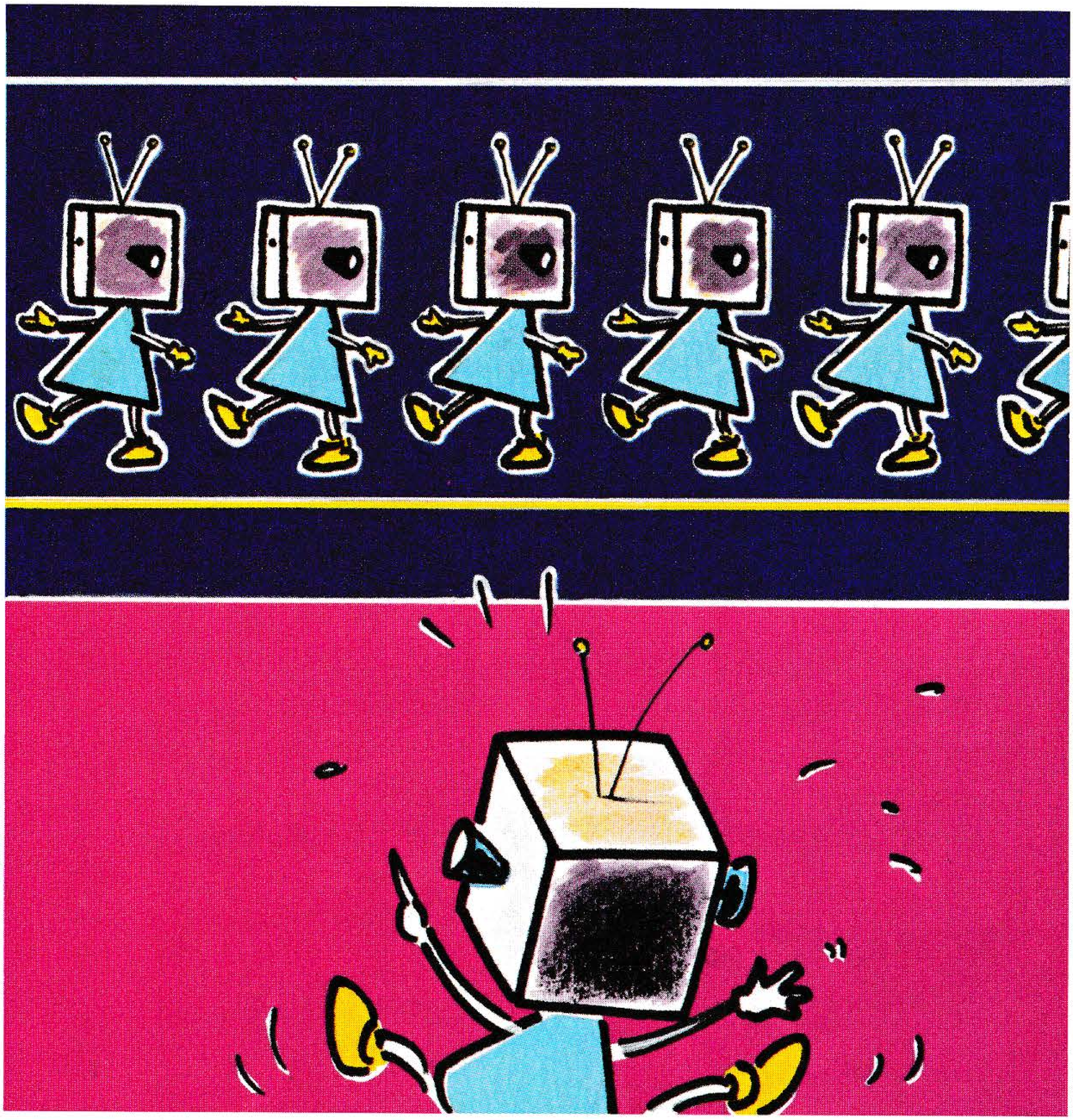
動くグラフィックのこころみ

- ●同じ形を並べて
- ●周期的繰返し
- ●移動するグラフィック
- ●キー入力による処理
- ●位置の判断
- ●スプライト機能を使おう

ここではグラフィックを取り扱うための基本的な方法や考え方についてお話いたします.

本書はそもそも文章による説明よりもプログラムを主役に据えようという意図のもとに書かれておりまして，何をどのように計算して各グラフィックが表示されているのか，それはプログラムを見て読者諸氏に判断していただこう，くだくだしい説明はやめるべし，と半ば教育的，半ばあなたまかせ的主旨が全体を貫いています（手を抜いたのでは決してありません．読者の方々にとってこれが最善の道であると我々は信じているのです）.

とはいえ，最初から摩訶複雑なグラフィック・プログラムをポンと皆さんの目の前に放り出し，すごいですねえ，さあ理解してくださいねえ，なんていうのはあまりに



無責任すぎるのではなかろうか？筆者としてそのソシリは免れ得ません．幾何学に王道はございませぬ，と見得をきった大数学者がいましたが，やはり学問知識というものはすべからく段階を追って基礎から積み上げていくことが肝要なのです．

そこで，本書ではこの２章において基礎的な図形の描き方を説明し，グラフィックのイロハに相当する部分をまず身につけていただこうというのです．ここで述べていることは3章以降のすべてのコンピュータ・グラフィックスの基本です．ぜひとも理解していただきたいものばかりであります．

位置を決める

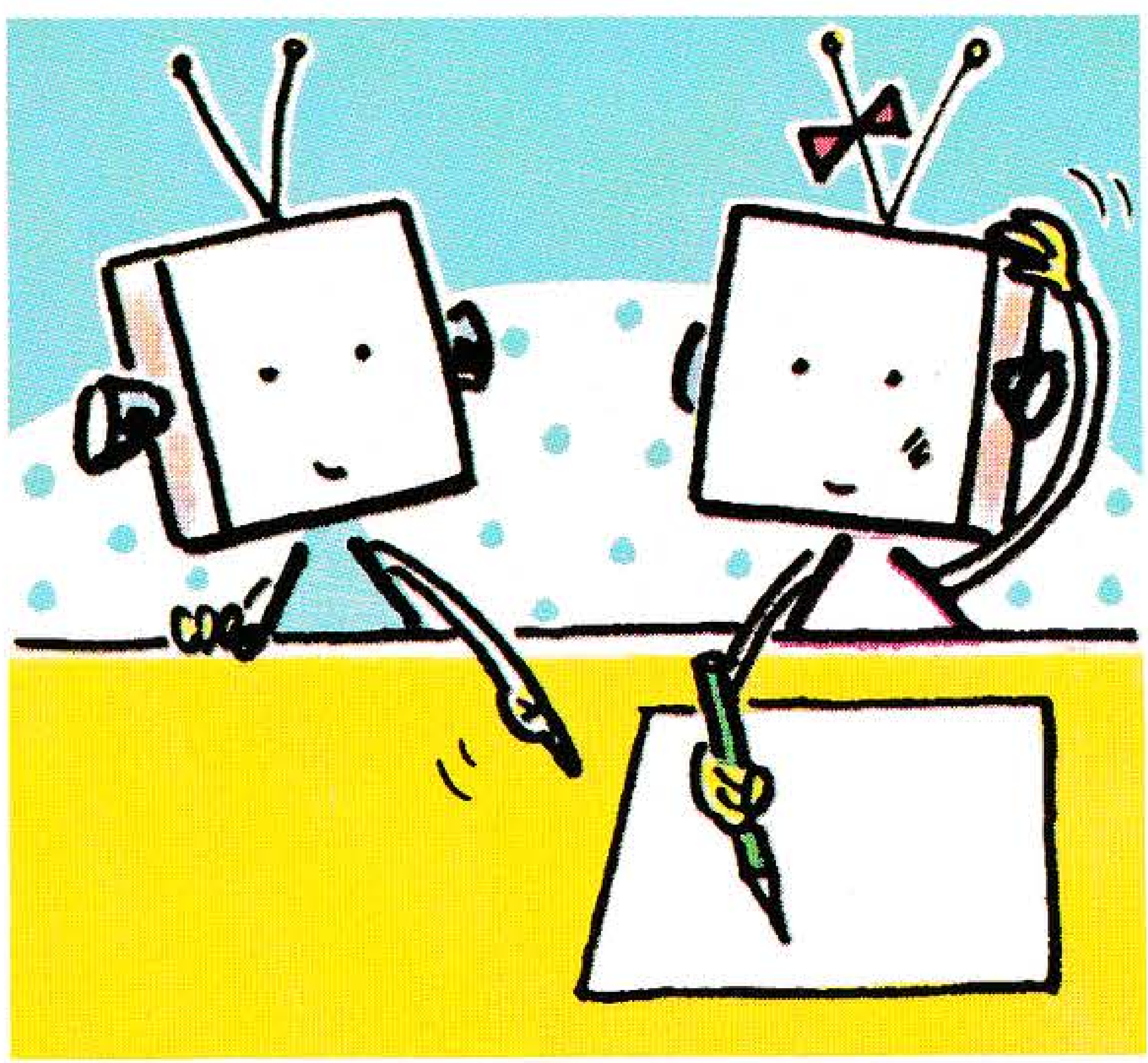
どこ？ ここはどこなの？ 少女がそうつぶやいたとき，天の一角から荘厳な声が聞こえてきた．

少女よ，心配するでない．そのまま百歩進み，そこから右手に二百歩進みなさい．おまえの住む世界へ通ずる出口がそこにあるはずだ．

少女は，この声に従って歩いていき，無事に両親の待つ自分の家へ帰ることができました．めでたし，めでたし……．

えっ！何をわけのわからんことをいっているのかって？ いえ実はこれからグラフィックの一番の基礎となる「位置の指定」についてお話しようと思いまして．

もうひとつ，これも例ですが，四角い紙の左上あたりに点を打つことを考えてみましょう．



しかもエンピツを持っているのがあなた以外の人物で，あなたは彼女あるいは彼に指示を与えて点を打たせるのだとします．その指示は，こんな具合になるのでしょうか．

あっそこじゃない．もっと上．その右，右だよ．あっ行きすぎだ………．

いえ，賢いあなたはきっとこんな指示の仕方はしないでしょう．そう，ただこういえばいいんですから．

上から2 cm，左から3 cmのところに点を打ってください．

これならひと言ですみます．

さて，神様の話と，四角い紙の話，この2つの位置指定の方法に共通していることは何でしょう．それは，「2個の数（データ）で位置を指定している」ということです．そう，平面上ではどんな点でも2個1組の数値によってピタリと位置が指定できるのです．

このように位置を指定するための2個1組の数値を座標といいます．コンピュータ・グラフィックスでは，この座標の使い方がまず基礎となります．それではこれから，MSX では実際にどんな座標を用いているのかみていきましょう．

MSXで用いる座標

MSX でグラフィックを扱うとき，2種類の画面モードを選ぶことができます．本章では，特に断わりがない限り高解像度グラフィック・モード（SCREEN 2）を基準に話を進めます．画面は，おおむね横4：縦3の比率の長方形になっています．この中が横方向256分割，縦方向192分割された，約4万9千個（256×192）のドットから画面が構成されています．ドットの形は，*Figure 1* のようにおおよそ正方形となっています．

これらの中から1個のドットを指定するために，次のような2組の数字を組み合わせた座標を用います．

(1)左端から数えて何個目のドットか．

このような横方向の位置を普通x座標といいます．

(2)上端から数えて何個目のドットか．

このような縦方向の位置を普通y座標といいます．

このx座標とy座標の2つでドットの位置を指定するわけです．たとえば *Figure 2* に示したドットは，左上端から5ドット右へ，2ドット下へ進んだ位置にあります．そこでこのドットの座標を（5，2）と表示します．

ここで，座標の値は0から始まることに注意してください．*Figure 3* のような画面の左上隅にあるドットの座標は，（0，0）なのです．（1，1）ではありません．

ついでにいえば，画面右下隅のドットは，座標（255，191）です．これも間違えやすいのですが，（256，192）ではありませんよ．

x座標が0～255，y座標が0～191，これがMSX で指定できる座標の範囲です．どんなグラフィックも，この範囲の中で表現されます．

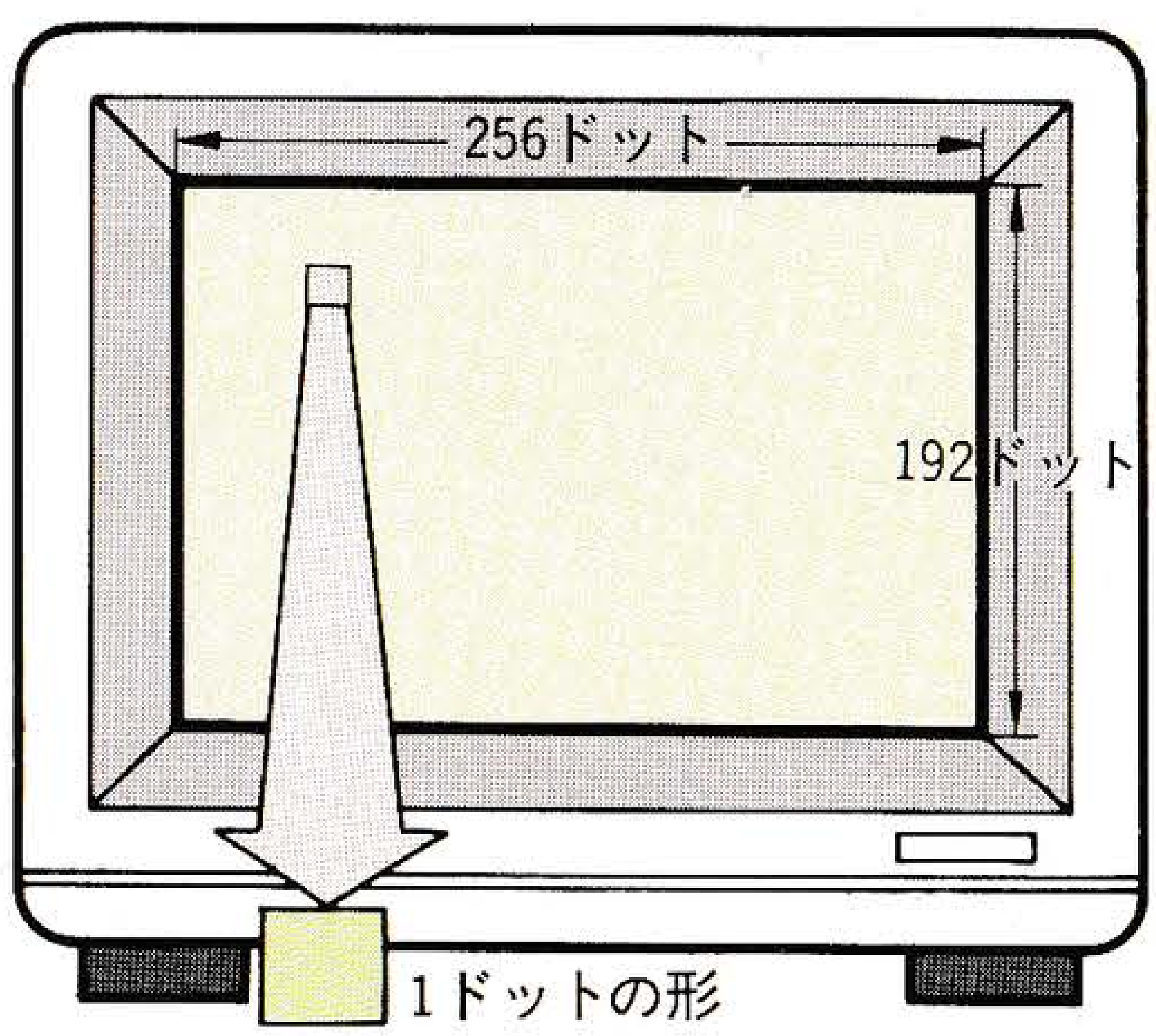


Figure 1

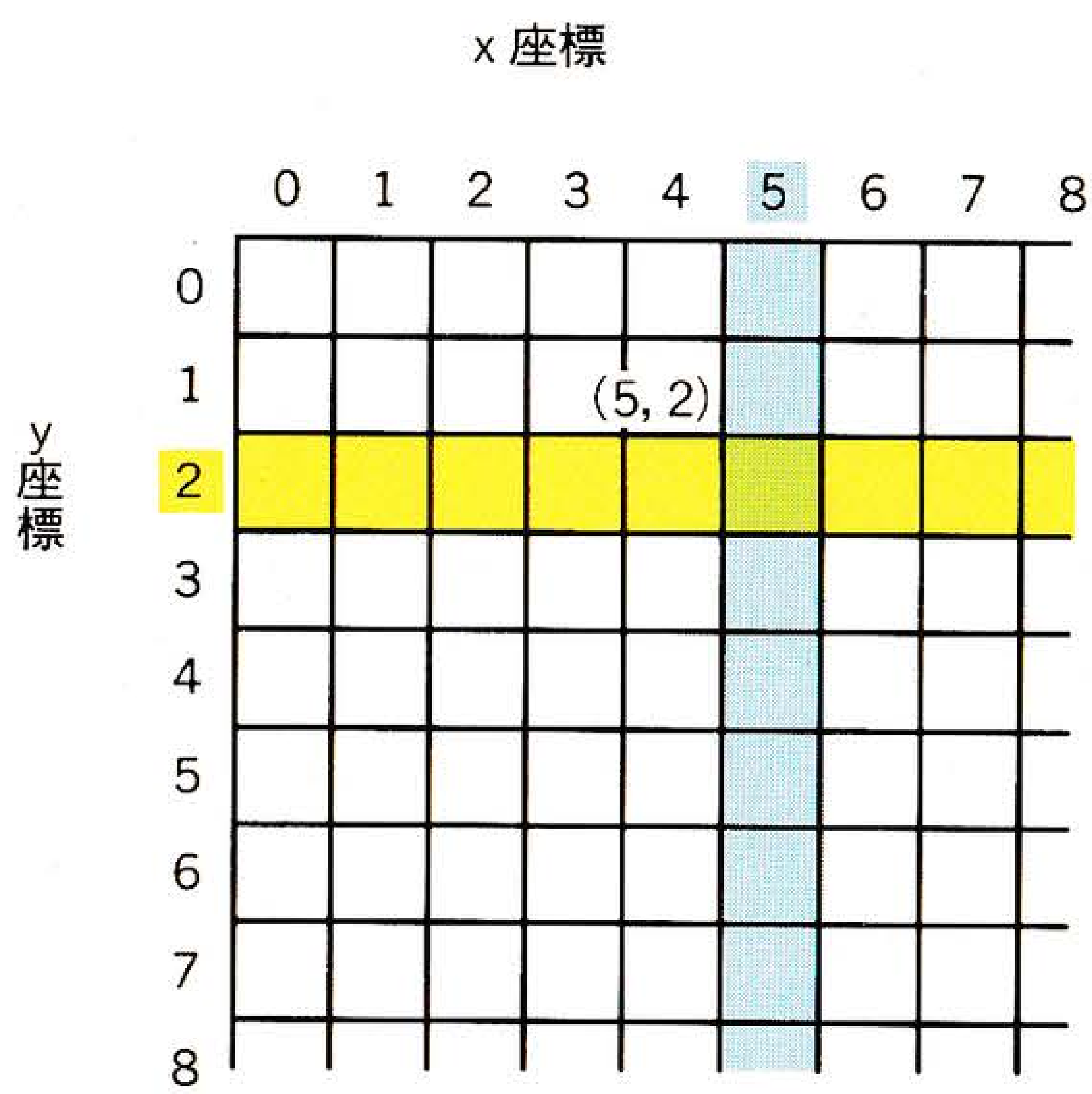


Figure 2

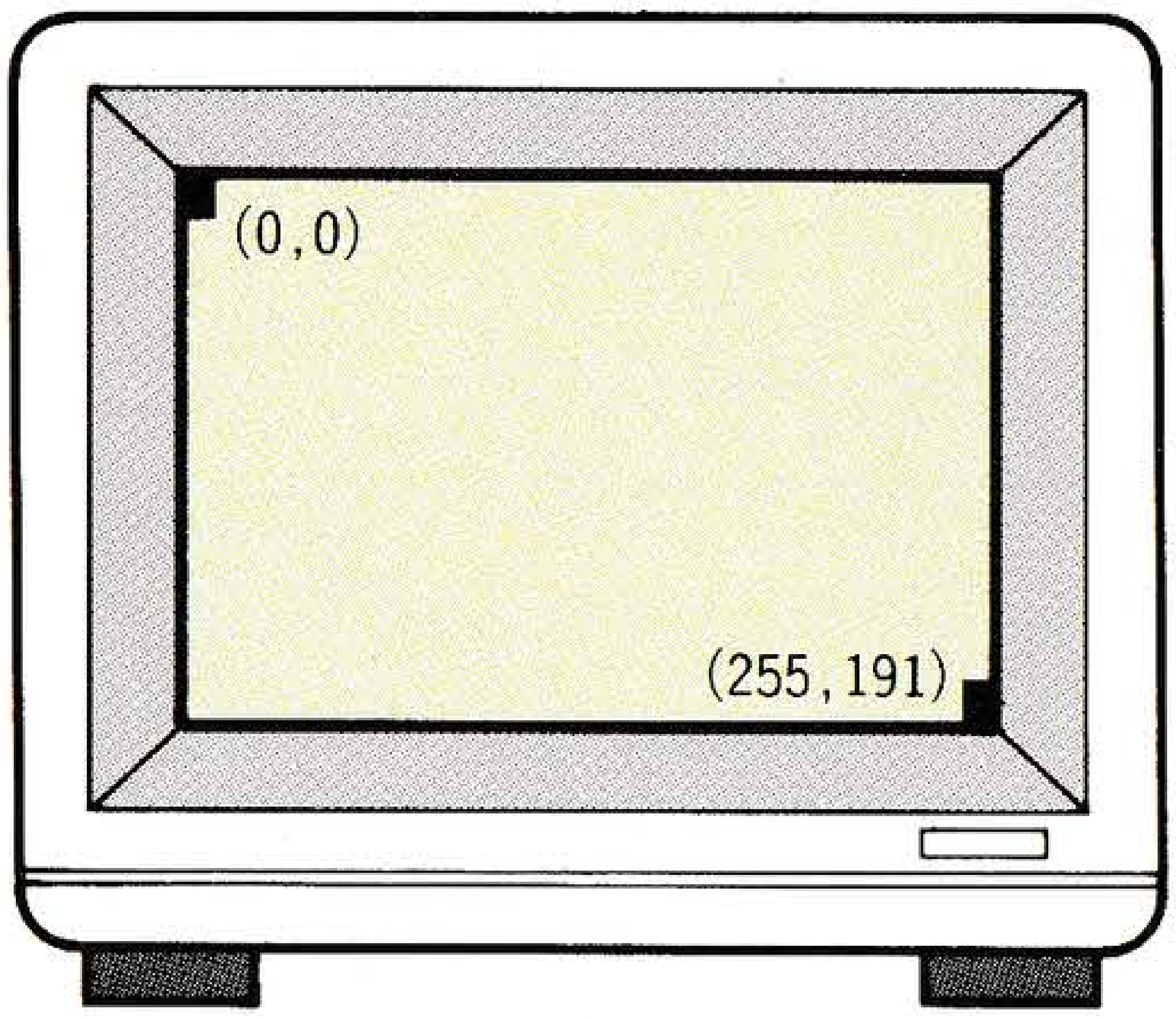


Figure 3

などというゴタクを並べるよりも，あなた自身の目で確かめるのが一番です．まずは *List 1* を打ち込んで実行してみてください．

150行の意味はわかりますか．これは座標（X0，Y0）の点と（X1，Y1）の点を結ぶ直線を引いているのです．ここでよく見ていただきたいのは，（X0，Y0）という点，そして（X1，Y1）という点の位置が画面のどのへんにあるかということです．

130行と140行がこの2点の座標を決めている部分ですが，X0=20，Y0=10ですから，

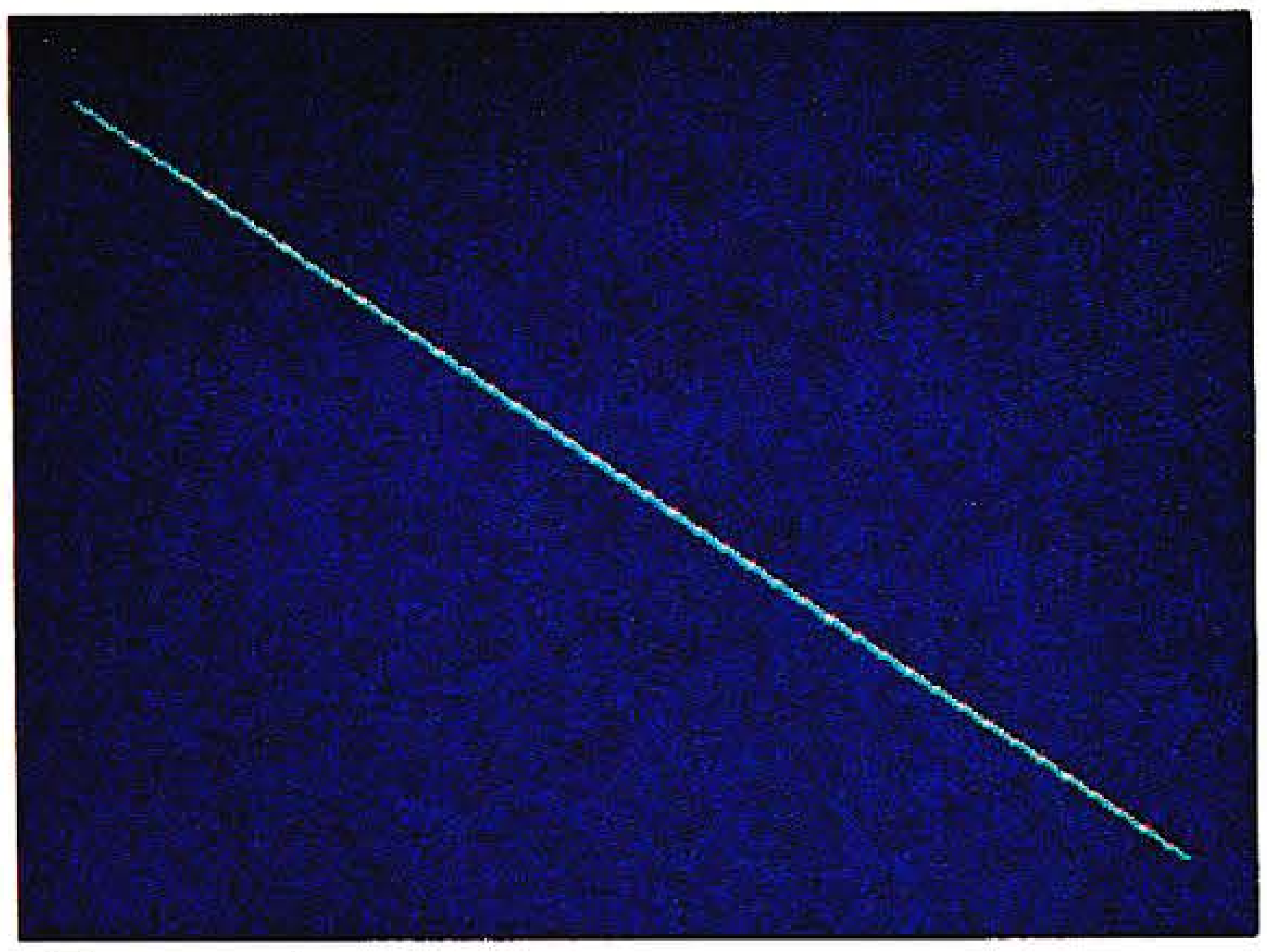
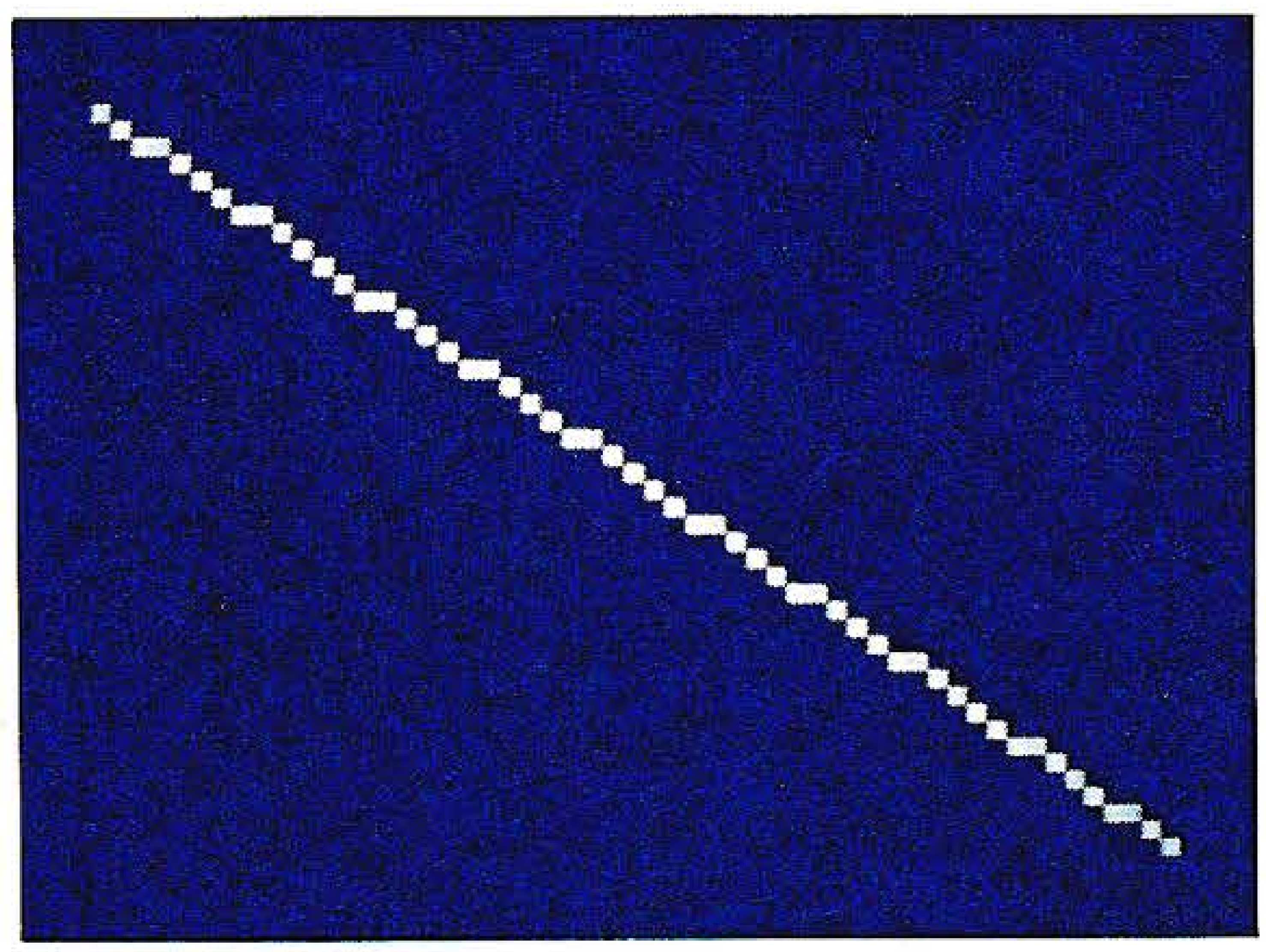


Figure 4a



b

List 1

```
100 ' List   1
110 '
120 SCREEN 2 : COLOR 15,4,4 : CLS
130 X0=20  : Y0=10
140 X1=235 : Y1=182
150 LINE (X0,Y0)-(X1,Y1),15
160 I$=INPUT$(1):END
```

List 1a

```
120 SCREEN 3 : COLOR 15,4,4 : CLS
```

はじめの点の座標は(20，10)，これは画面の左上あたりにあります.終わりの点は(235，182)で，これは右下あたりの座標です．それで *Figure 4a* のような直線が引かれるのです．よろしいですか?

130行や140行で指定する数値を変えて，いろいろな点の位置と座標の関係をつかんでください．たとえば，*Figure 4a* とは逆に，右上から左下へ線を引くには2点にどんな座標を与えればよいでしょうか．横にまっすぐ線を引くには? 垂直な線は描けますか? はい，さっそく挑戦してみましょう．

座標の値を変化させるときには，x座標の範囲が0〜255，y座標の範囲が0〜191であったことに注意してください．この範囲を超えると何が起きるかって? それは自分でためしてください．

なお160行は，プログラムが終了してしまい，テキスト画面に戻りグラフィックを壊してしまうのを防いでいます．プログラムは何かキーを押すと終了します．

次にマルチカラーモード(SCREEN 3)についてですが，説明の前に *List 1* の120行を *List 1a* のように変更後，実行してください(*Figure 4b* 参照)．

どうですか?線がひどくガタガタになってしまったでしょう．これは，マルチカラーモードで描くことができるもっとも小さな領域(ブロック)が，高解像度グラフィックモードの4×4ドット分に相当するからです．このマルチカラーモードでは，1ブロック，1ブロック，自由に色が変えられるかわりに，グラフィックはブロックを最小単位として描かなければなりません．1画面は縦48ブロック，横64ブロックで構成されています．しかし座標に関しては，ドット単位のままです．1ブロックが16ドットで構成されているため1ブロックの座標も16ドット分存在することになります．しかし，これ

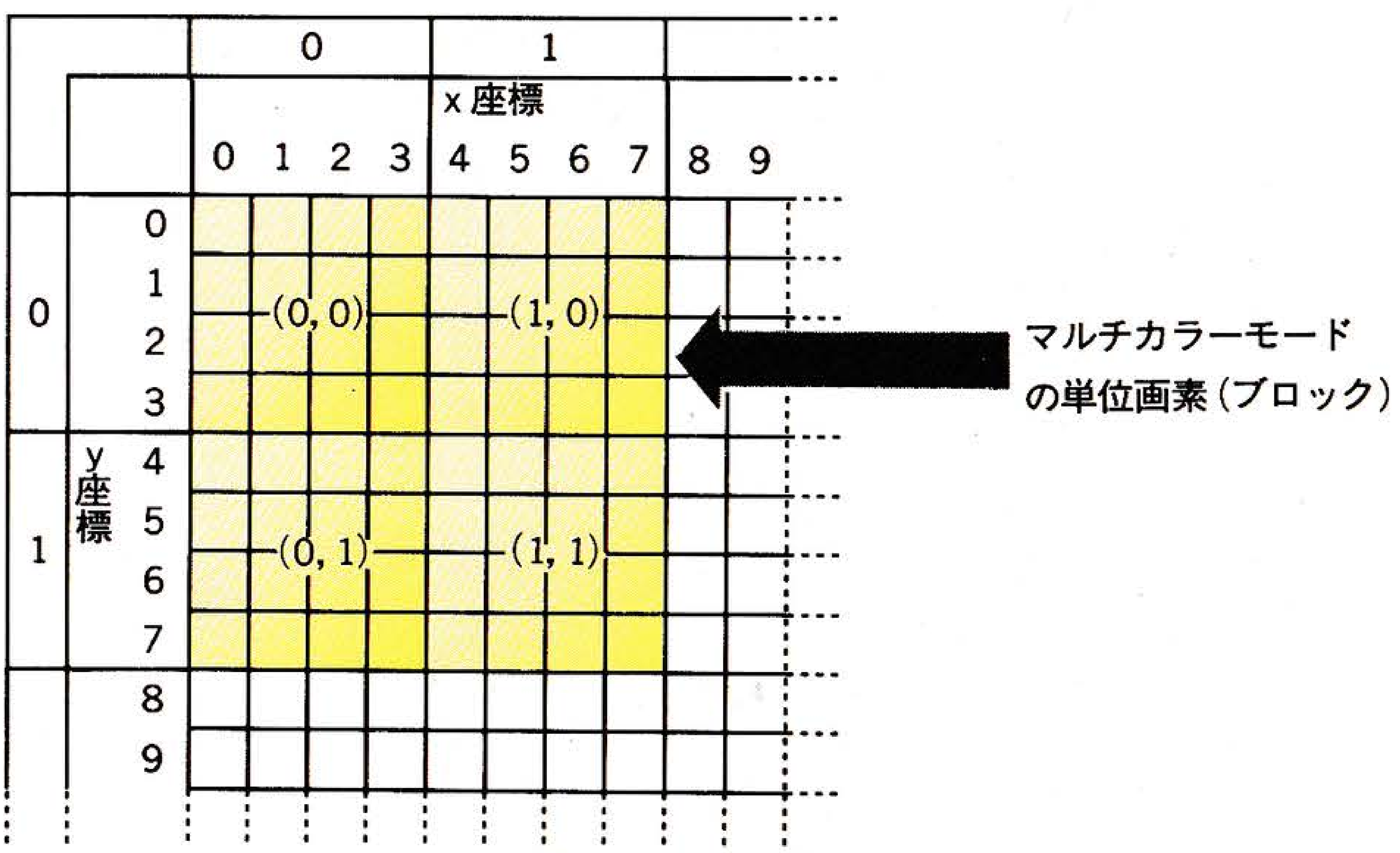


Figure 5

ら座標のどれでもブロックの座標として用いることができます．

たとえば，*Figure 5* で左すみに点を打ちたい場合，左すみのブロックは，(0，0)から（3，3）まで16の座標を持っているので，このうちのどれか1つを選び，PSET(2，2)とかPSET(0，0)などを実行すれば，左すみの1ブロックを点として表示します．

中間点を求めて

座標を利用すると，いろいろなおもしろいことができるのですが，その中からいくつか簡単なものを取り上げて紹介していくことにします．最初は「2点の中間の位置を求める」という例から．

ここで，「2点のちょうど真ん中の位置を指しなさい」といわれたら，あなたはどうやってその位置を探すでしょうか．そう，たとえば，「日本地図で東京と大阪の中間はどこですか？」と聞かれたりしたらー．

ええと，東京と大阪をこうやって直線で結んで．この長さを測ると○cm．その半分の□cmの場所は……，わかりました×××でーす（*Figure 6* 参照）．

こうやるのも一案ではありますが，ここで座標を使って中間点を計算するたいへんスマートな方法を紹介しましょう．

えっ？ 東京や大阪にも「座標」があるのかって!?

はい，立派な座標がございます．それを緯度と経度と申します．

この地球上のどこを捜しても，緯度経度によって指定できない土地はありません．東京しかり，大阪またしかり，地図帳を見てみると，東京は北緯35.66度，東経139.78度．そして大阪は北緯34.70度，東経135.50度となっています．ですからこの中間地点

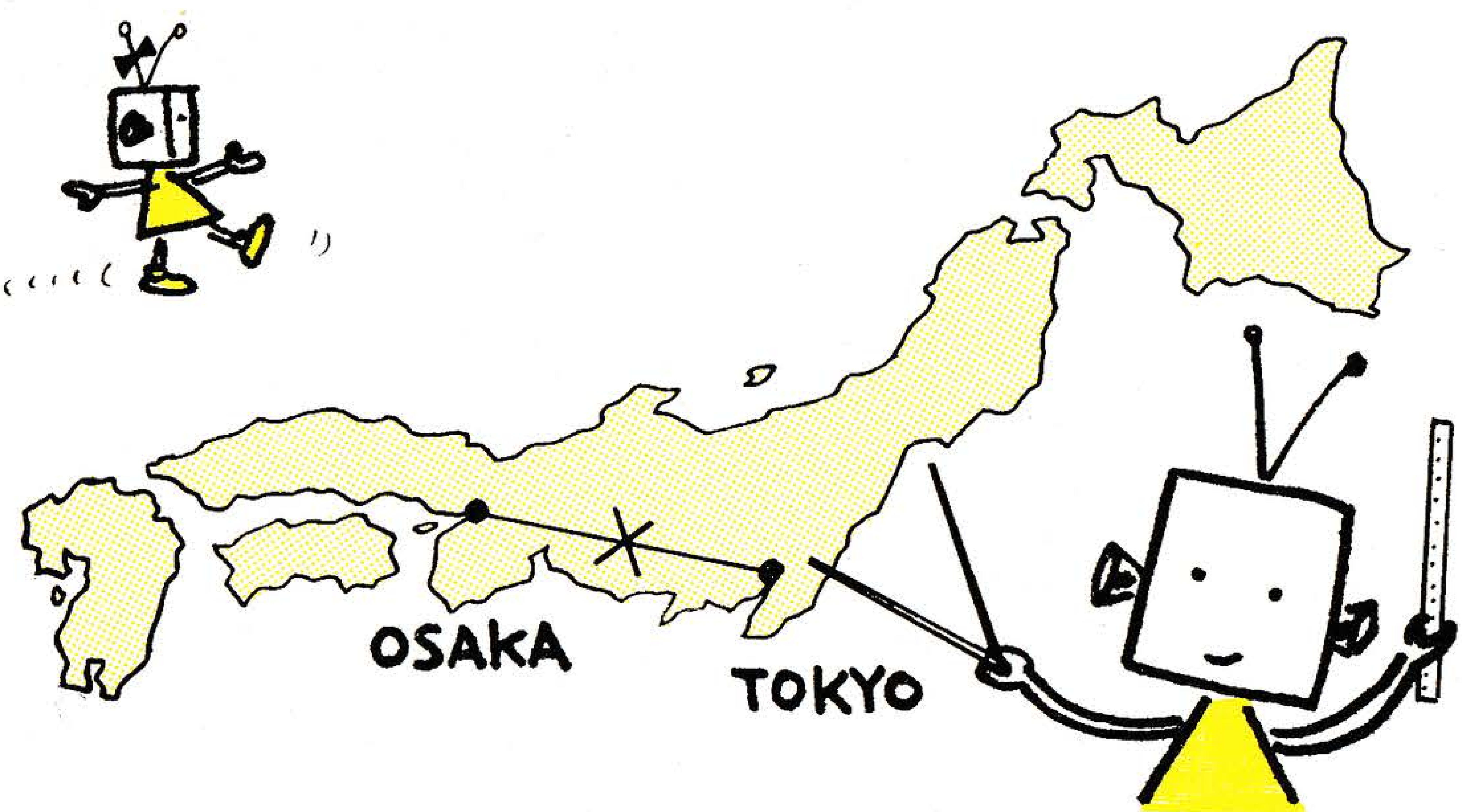


Figure 6

は北緯35.18度，東経137.64度すなわち愛知県と長野県の境に位置する，その名も茶臼山であることがわかります．地図にモノサシを当てる必要も，エンピツで印をつける必要もなく，簡単な計算だけでこの結果が得られたのです．

これはどうやって計算したのかと申しますと，まず緯度は緯度同志足して2で割ります．

(35.66＋34.70)／2＝35.18

この値が中間地点の緯度になります．同様に経度も経度同志で計算すると次のようになります．

(139.78＋135.50)／2＝137.64

さて，いよいよコンピュータ・グラフィックスの話に入ります．実は，グラフィックで2点の中間を求めるのもまったく同様に行います．2つの点があったら，そのx座標同志を足して2で割ったものと，y座標同志を足して2で割ったものが2点の中間の座標となるのです．

List 2 がこれを応用したプログラムです．130，140行で，2点の座標を乱数で決めます．次の150行がお目当ての部分．ここでは，(X0，Y0)という点と(X1，Y1)という点の中間の位置を求めて(X2，Y2)としています．

160，170行は本当に中間点が求められたのかを目で確かめようというところ．160行で描いた円を170行の直線がつらぬく様子を見てください(*Figure 7* 参照)．なおこ

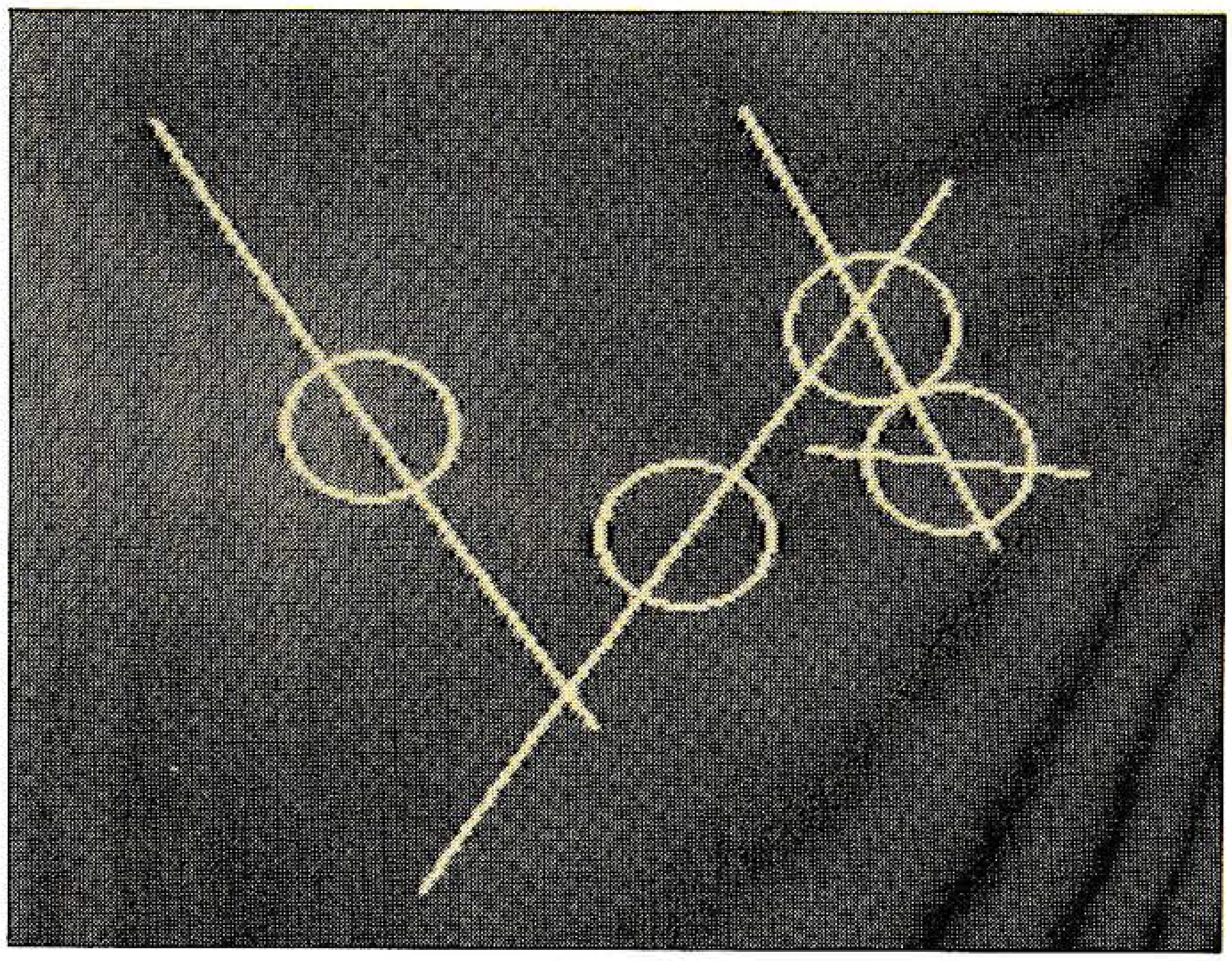


Figure 7

List 2

```
100 ' List   2
110 '
120 SCREEN 2 : COLOR 15,13,13 : CLS
130 X0=RND(1)*256 : Y0=RND(1)*192
140 X1=RND(1)*256 : Y1=RND(1)*192
150 X2=(X0+X1)/2  : Y2=(Y0+Y1)/2
160 CIRCLE(X2,Y2),15
170 LINE (X0,Y0)-(X1,Y1),15
180 GOTO 130
```



このマークが付いているプログラムは CTRL ＋ STOP で終了させてください．

のプログラムは，CTRL＋STOPで終ります．

さてここで話を一歩先へ進めましょう．座標を使って2点のちょうど真ん中の位置が計算できることは，これでおわかりいただけたと思います．けれど，いつも中間点ばかりが必要になるとは限りません．ときには端っこから3分の1の点や4分の1の点が欲しい場合もあります．

その場合の計算方法は次のとおり．いいですか，一気にいいますからね．

点(X0，Y0)と点(X1，Y1)を結ぶ直線の上で，始点(X0，Y0)から測って直線の長さのM分のNにあたる点の座標(X2，Y2)を求める式は，

```
X2=((M-N)×X0+N×X1)/M
Y2=((M-N)×Y0+N×Y1)/M
```

となります(*Figure 8* 参照)．

これでは何だかよくわからないでしょうから実例を挙げてみます．始点の座標を(20，10)とし，終点の座標を(220，180)とします．そして始点と終点を結ぶ直線上で，始点から5分の2離れた位置は，どこでしょうか？

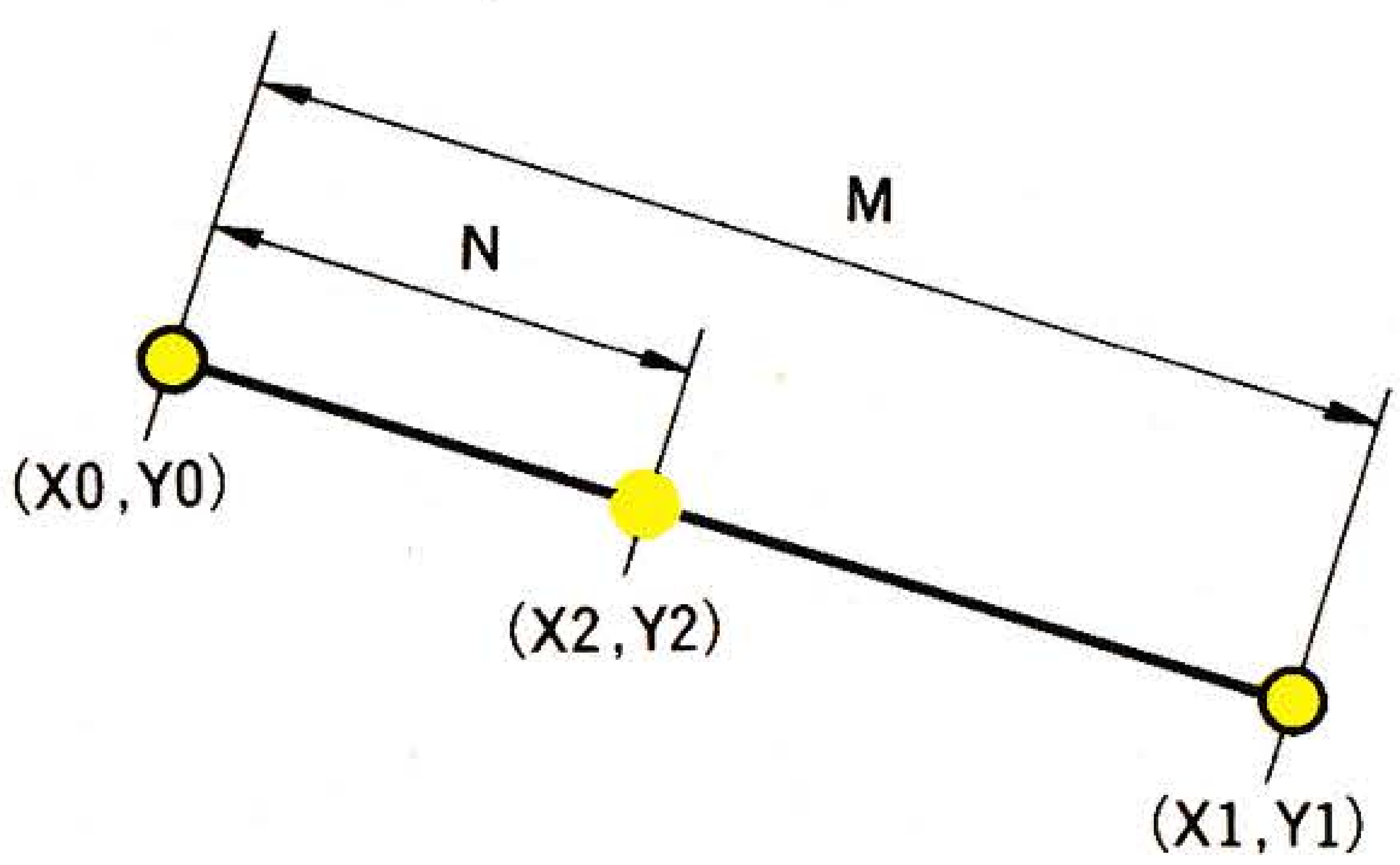
この場合は，先ほどの式で，

```
X0=20, Y0=10, X1=220, Y1=180,
M=5, N=2
```

とおいて計算すれば，

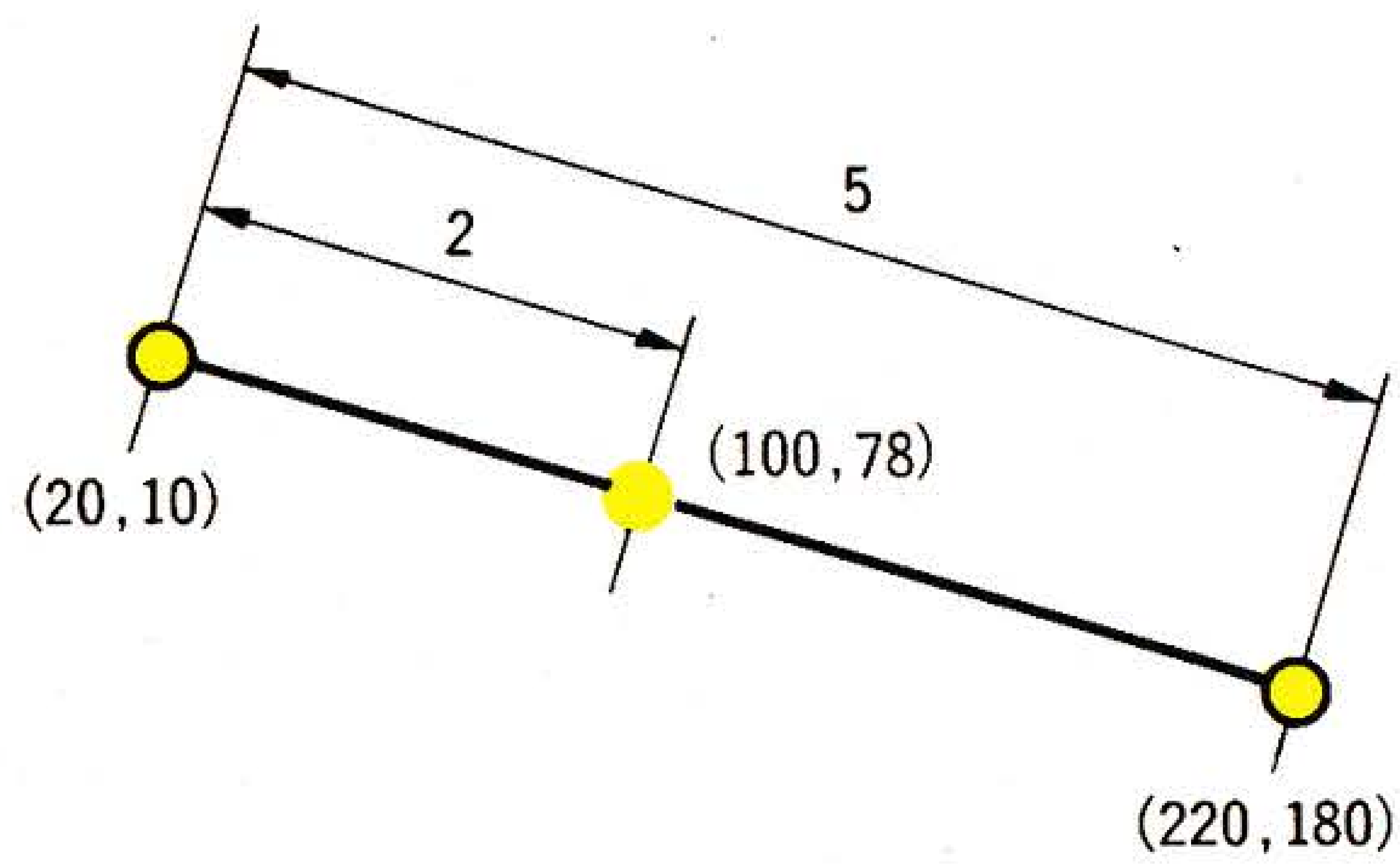
```
X2=((5-2)×20+2×220)/5=100
Y2=((5-2)×10+2×180)/5= 78
```

つまり，(100，78)であると計算されます．要領はわかりますよね(*Figure 9* 参照)．



$$X2=((M-N)\times X0+N\times X1)/M$$
$$Y2=((M-N)\times Y0+N\times Y1)/M$$

Figure 8



$$X2=\frac{(5-2)\times 20+2\times 220}{5}=100$$
$$Y2=\frac{(5-2)\times 10+2\times 180}{5}=78$$

Figure 9

この計算式を使ったプログラムが，*List 3* です．これは直線を10等分して，端から順に印をつけていきます．180行と190行が直線を10分のNに分ける点の座標を計算し，200行でその位置に円を描きます．これをN=0からN=10まで繰り返すわけです（*Figure 10* 参照）．ここで，N=−1とかN=11とかの値を与えるとどんな結果になると思いますか?

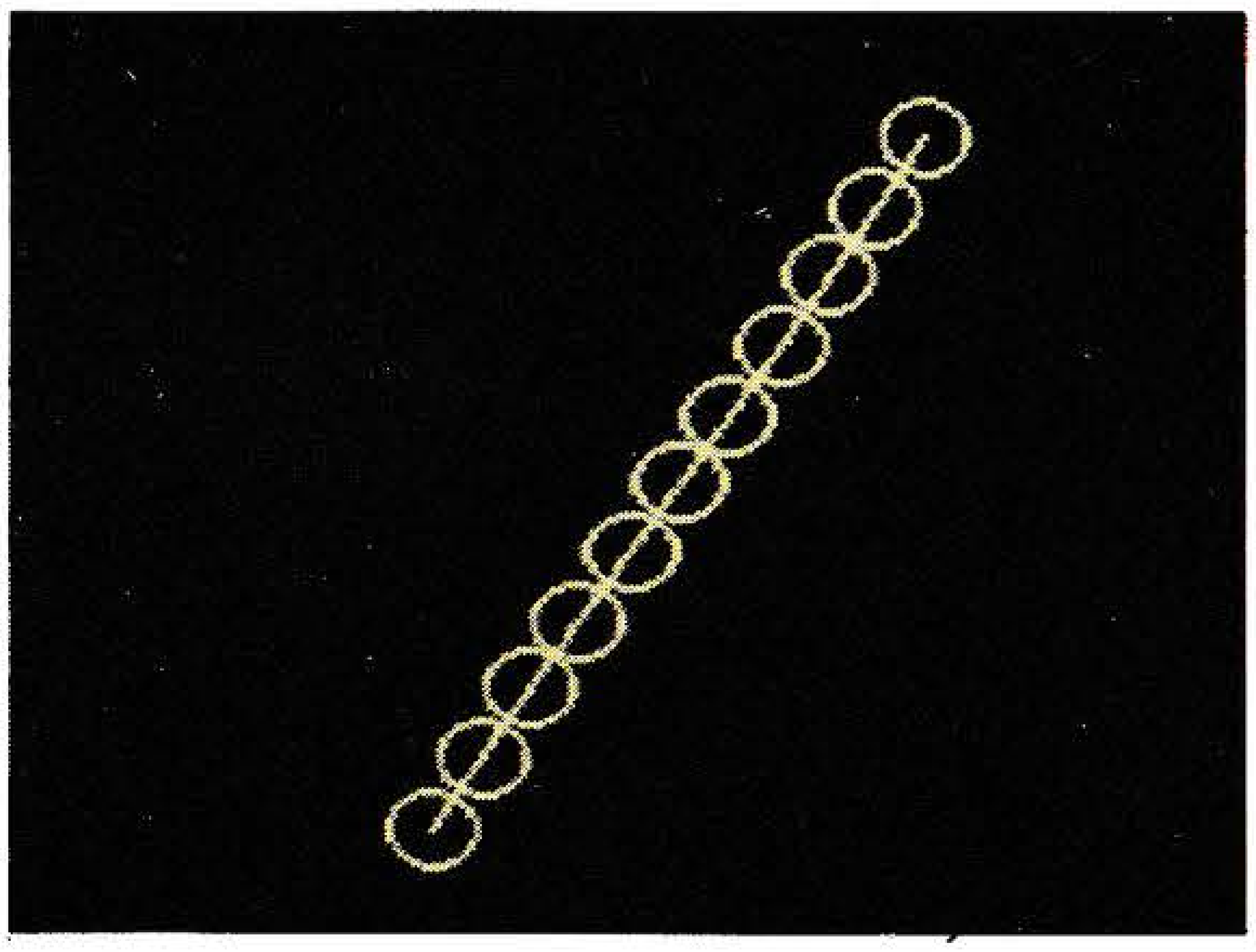


Figure 10

図形の移動

座標を使ってできることの第2弾．図形の位置を動かすための方法を説明します．複雑で華麗なグラフィックももとをたどればここに示すような小さなことを積み重ねてプログラミングされているのです．

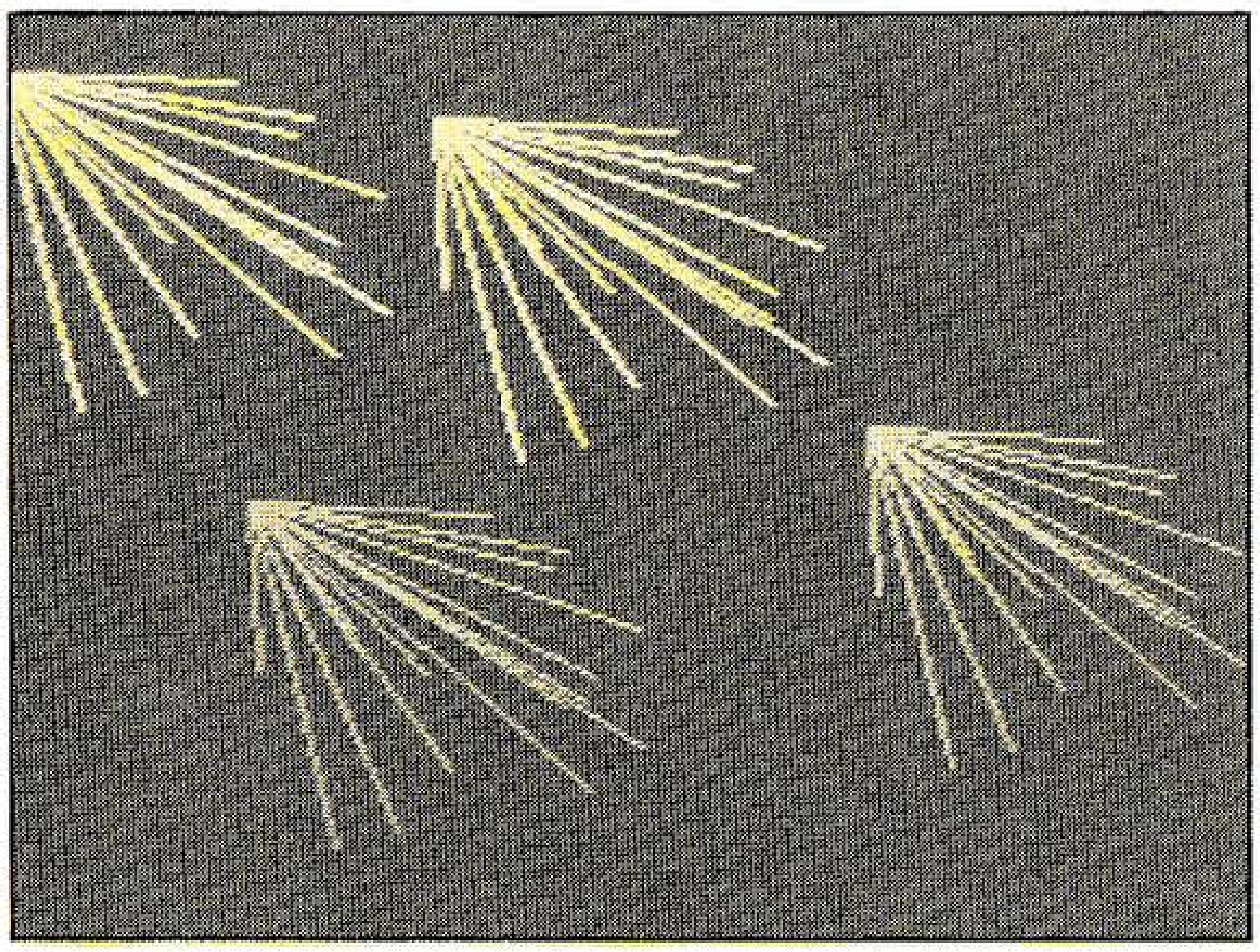
まず図形の向きや形を変えないで，位置だけ移す方法．これは図形における各点の座標に一定の値を足す（または引く）だけで行えます．

図形を N ドット右へ移すなら，x 座標に N を足し，逆に左へ移すなら x 座標から N を引きます．また，y 座標に N を足せば図形は下方へ N ドット移動し，N を引けば上方へ N ドット移動します．

たとえば，（X，Y）という座標に描かれた図形を，右へ 85ドット，下へ10ドット移動させた場合の，新しい座標は(X+85，Y+10）です．

List 3

```
100 ' List  3
110 '
120 SCREEN 2 : COLOR 15,2,2 : CLS
130 X0=RND(1)*256 : Y0=RND(1)*192
140 X1=RND(1)*256 : Y1=RND(1)*192
150 LINE (X0,Y0)-(X1,Y1),15
160 '
170 FOR N=0 TO 10
180   X2=((10-N)*X0+N*X1)/10
190   Y2=((10-N)*Y0+N*Y1)/10
200   CIRCLE (X2,Y2),8,15
210 NEXT N
220 '
230 GOTO 120
```



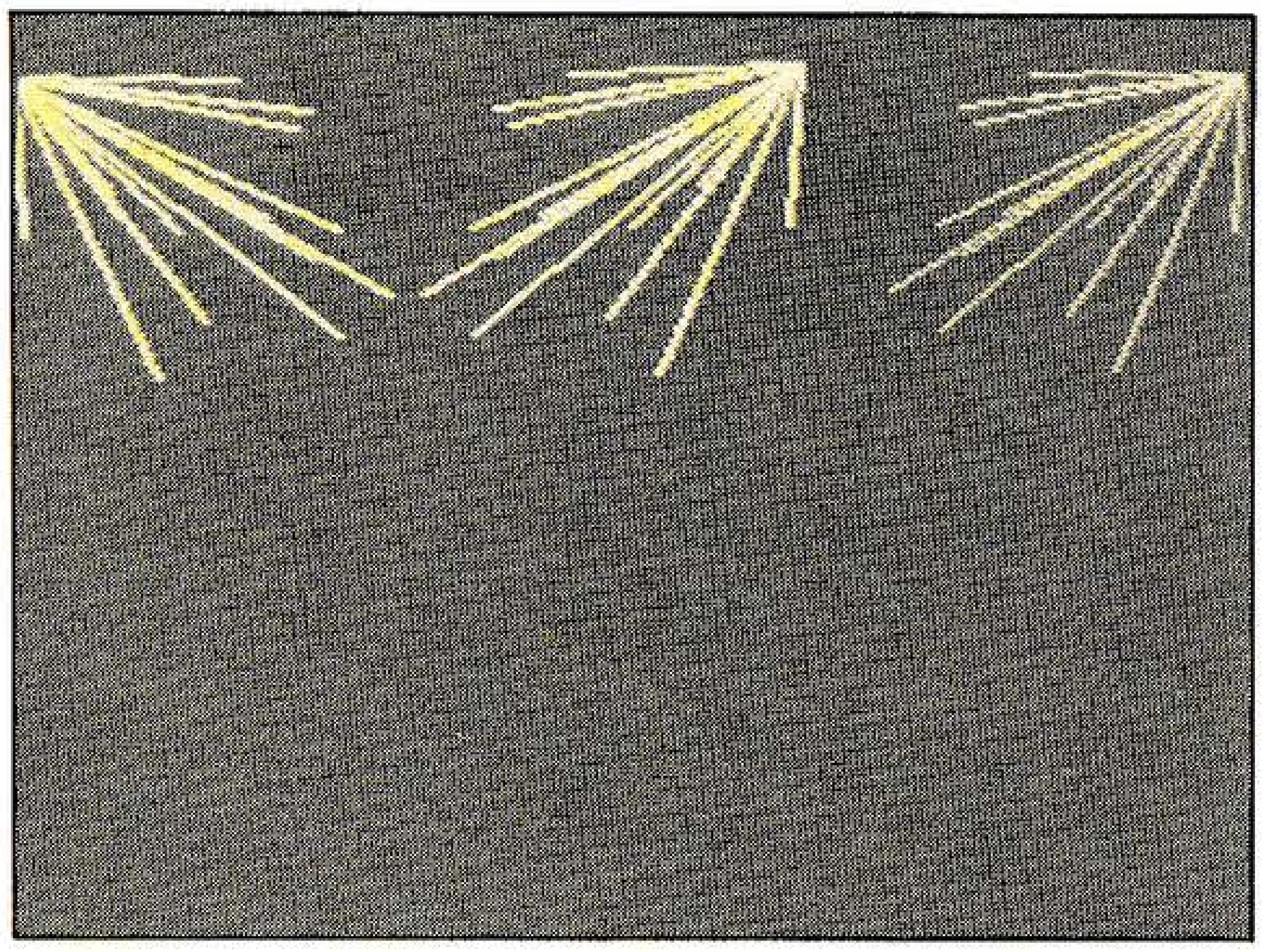


Figure 11 (List 4の実行結果)　　(List 4aの実行結果)

List 4

```
100 ' List  4
110 '
120 SCREEN 2 : COLOR 15,4,4 : CLS
130 X=RND(1)*80 : Y=RND(1)*80
140 LINE (0,0)-(X,Y),15
150 '
160 LINE (85,10)-(85+X,10+Y),11
170 LINE (170,80)-(170+X,80+Y),9
180 LINE (50,100)-(50+X,100+Y),2
190 '
200 GOTO 130
```

List 4a

```
160 LINE (160,0)-(160-X,Y),11
170 LINE (255,0)-(255-X,Y),9
180 REM
```

List 4b

```
160 LINE (0,191)-(X,191-Y),11
170 LINE (255,191)-(255-X,191-Y),9
180 REM
```

これを *List 4* のプログラムで確かめてみましょう．140行で引いた直線を基本図形として，それを右へ85ドット，下へ10ドット全体的に移したものが160行で描いている直線です（*Figure 11* 参照）．

それでは170行と180行で描いている直線は，基本図形をどれだけ移動させたもの？答は，書かなくてもいいね．

線の終点（X，Y）だけではなく，始点の（0，0）もちゃんと移動させていることに注意してください．図形を構成している全部の点を同時に増減することが肝心なのです．

ちなみに，このように向きも大きさも変わらない図形の移動は，「平行移動」と呼ばれております．

こんな「平行移動」は，まだ単純，もっと歯ごたえのあるやつはないのかとおっしゃる方もいるでしょう．だんだんむずかしくなっていきますから，ご安心を？

次に説明したいのは，図形を裏返しにする方法，これは左右の逆転と上下の逆転の2種類あります．

左右を裏返しにするには，x 座標にマイナス符号をつけます．つまり，（x，y）という座標に対して（−x，y）を計算すればよいのです．こうすると，本のページを右から左にパタンとめくるように，図形の左右の向きがひっくり返ります（*Figure 12* 参照）．このとき本の背に当たる位置つまり移動の中心となるのが，画面でいえば左端，座標でいえば x=0の地点です．

ただし，ひっくり返した（−x，y）の位置は，このままでは画面の外へ出てしまうので，全体を右方向へ平行移動してやらなければなりません（*Figure 13* 参照）．それではどれだけ移動させればよいのか．この際，知っていると便利なのは次の法則です．

座標（−x+（2×a），y）の位置にある図形は，座標（x，y）の図形をx=aの位置を中心として左右裏返しにしたものである．



Figure 12

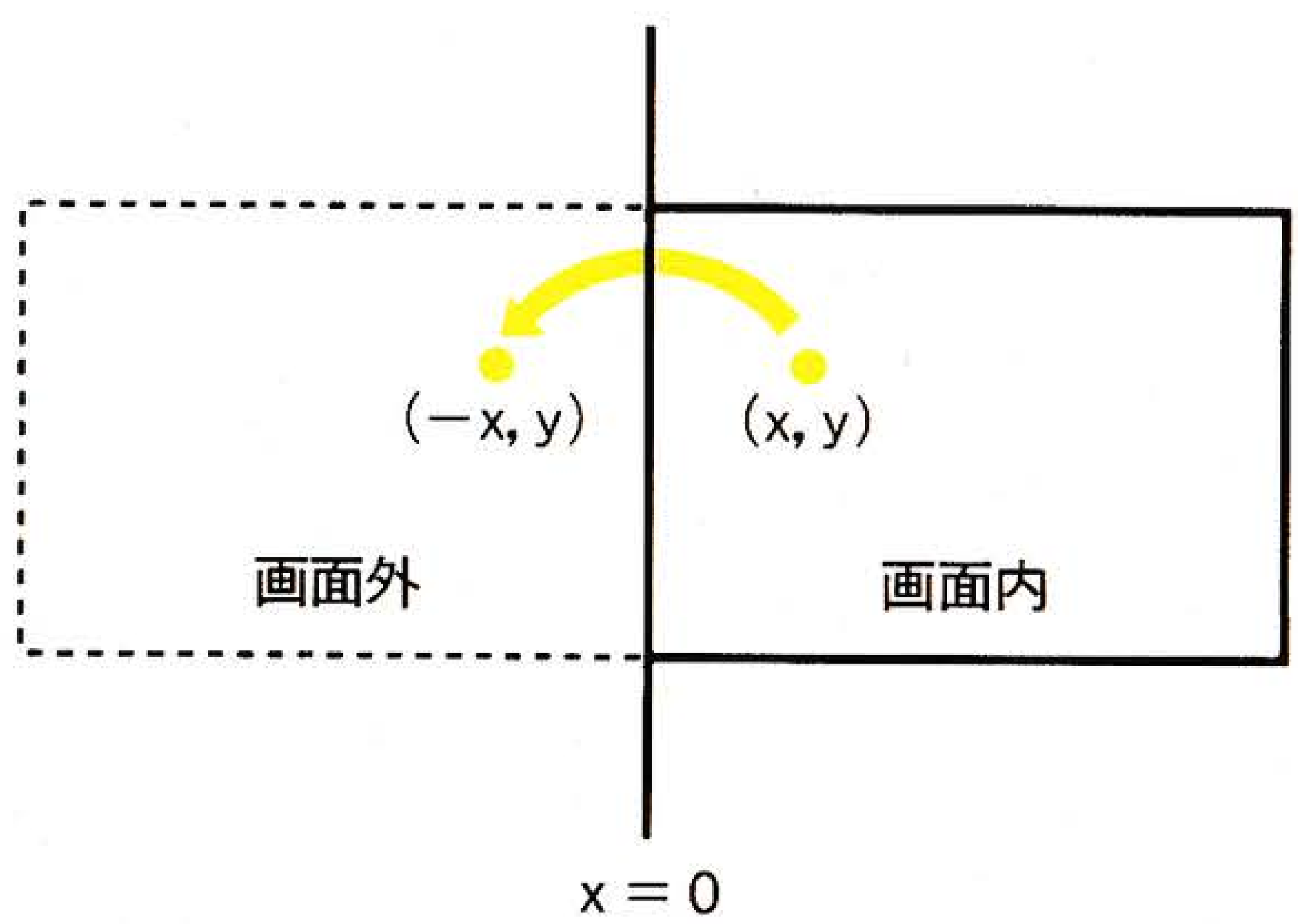


Figure 13

つまり，x 座標をマイナスしてさらに右へ2×a ドット平行移動させると，それは結果として x=a を中心に左右を逆転した図形になるのです（*Figure 14* 参照）.

これを確かめるために，*List 4* の160〜180行を *List 4a* のように変えて実行してみてください.

140行で描いた直線を基本図形として，x=80の位置を中心に左右ひっくり返したものが160行です（160−x というのは−x+160と同じことですよ，念のため）.

170行のように，x 座標を255−x として計算すると，ちょうど画面を裏返しにした形となります.

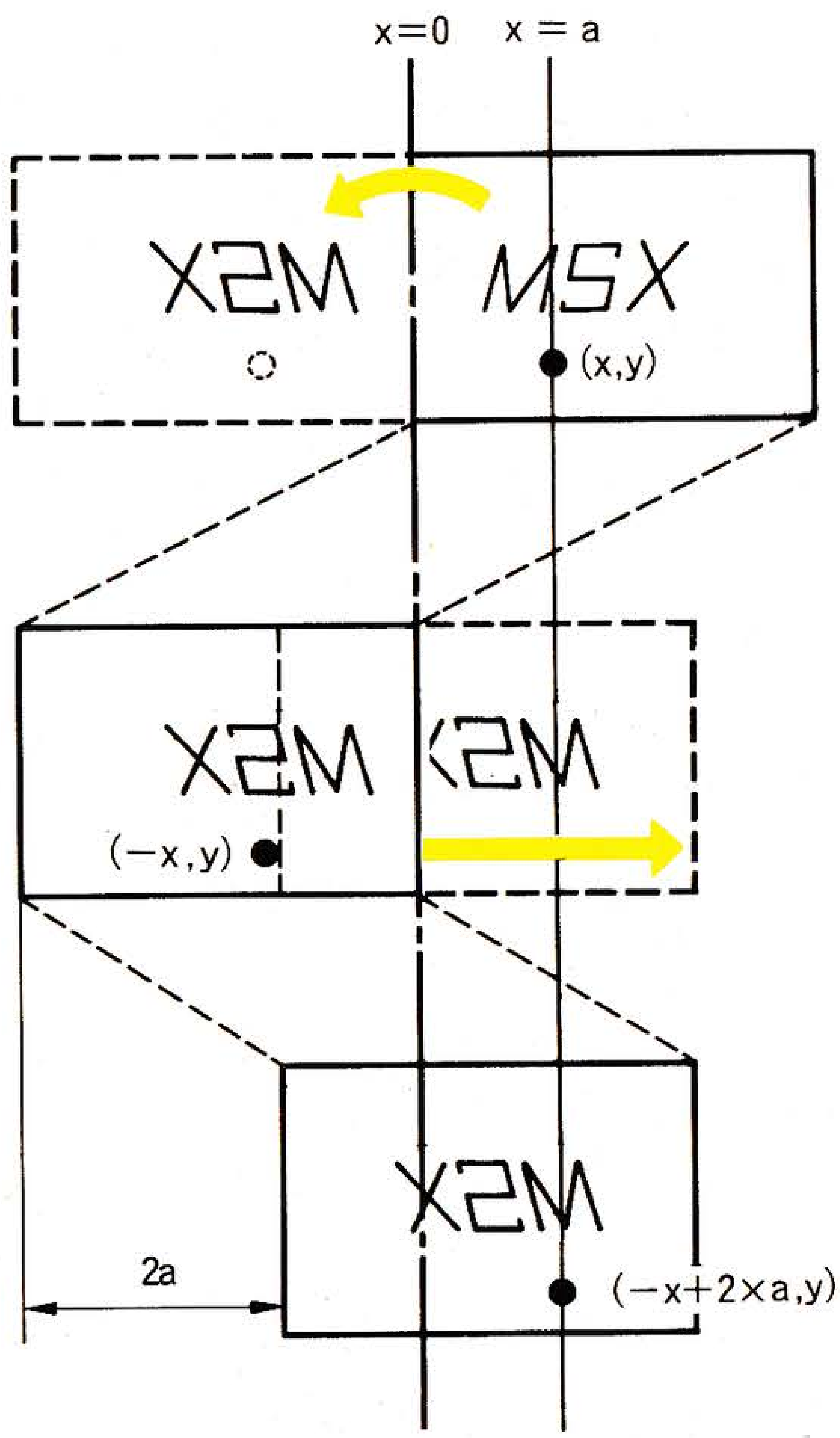


Figure 14

さて次に，上下を裏返すにはどうするか．これはy座標について，さきほどと同じ操作を行います．つまり，座標（x，y）の図形をy=bの位置を中心として上下を裏返した図形は座標（x，−y+(2×b)）によって計算できるというわけです.

さらに，*List 4* の160〜180行を *List 4b* のように変えてみてください.

160行が上下の裏返し，そして170行は上下，左右の両方とも裏返しています.

キャラクタ座標

ここまでは，すべてグラフィック座標についての説明を行ってきましたが，MSX で使用する座標にはもう一種類，キャラクタ座標があります．これも簡単に説明しておきましょう.

SCREEN 0 や SCREEN 1 によって設定されたテキスト画面で，文字や数字などを表示する位置は，キャラクタ座標によって指定します．このキャラクタ座標は，

SCREEN　0のとき
　x座標0〜39，y座標0〜23
SCREEN　1のとき
　x座標0〜31，y座標0〜23

となっています．ただし，ファンクションキーの内容が表示されている場合，y 座標の23は当然使えません．また WIDTH 命令で横一列の文字数を制限している場合は，x 座標の範囲が制限されます.

「絵」をつくる

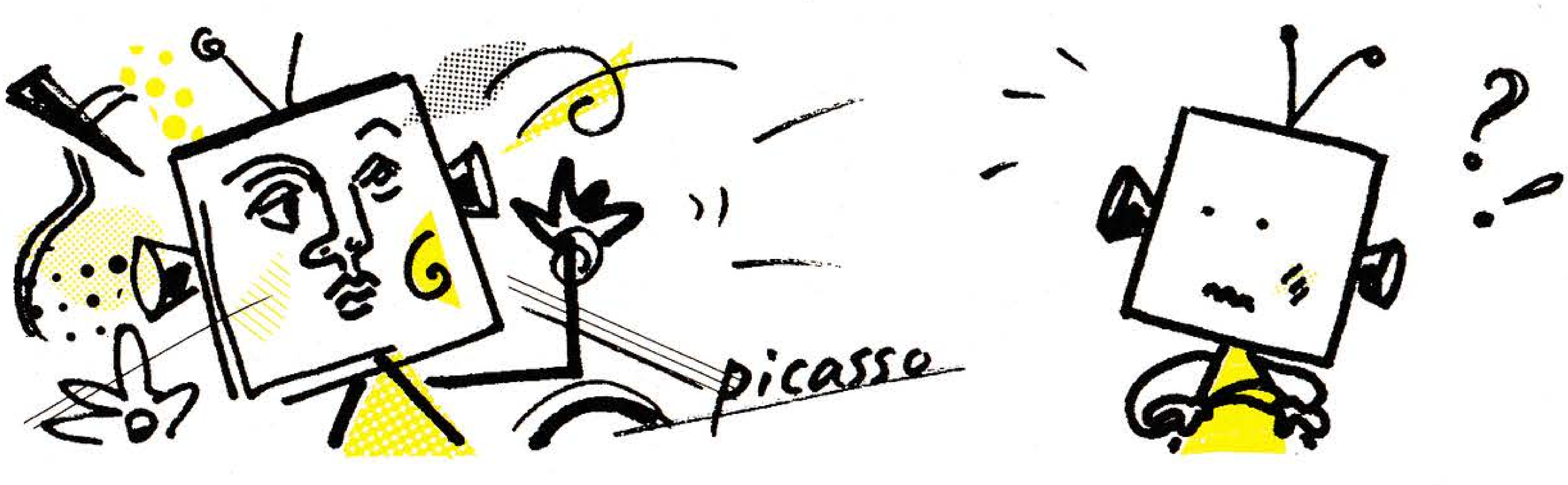
……というわけでありまして，前節では主に「座標の取扱い」ということで話を進めました．座標といえば，これはもう中学，高校で苦しめられた（あるいは今現在，苦しめられている真っ最中の）数学のことを思い浮かべる方も多いことでしょう．そうです，前節で私達は数学していたのです．その冷たくて味もそっけもない非人間的かつ学生すべての心労のもと？ である数学の世界から離れ，ここでグラフィックの芸術的可能性を眺めてみましょう．

本節で問題になるのは数字を扱う能力よりも，むしろいかにして美しい絵を描き出すか（ほら，同じグラフィックの表示結果でも前節では「図形」といっていましたが，今や「絵」といい換えられました），ようするにあなたの絵心，美的センス，といったものが大切になってきます．

あっ，不安そうな顔をしている人がいますね．なに 図画工作は昔から2しかもらったことがないって？ いえ，そんな心配はご無用．コンピュータ・グラフィックスのよいところは，手先の器用さが作品のできばえに関係しないという点です．器用な人が鮮やかにタイプしたものであろうが，ブキッチョな人が打ち込もうが，プログラムさえ同じであればMSXは常に同じ動作をしてくれます．

今まで絵なんて描いたことないよ，という人にとっても，コンピュータ・グラフィックスは恰好の道具として，そして大いなるインスピレーションの源となって，きっとあなたの眠れる感性を引き出してくれるでしょう．

と，話はとてつもなく大きくなってしまいましたが，要は，せっかくのコンピュータ，きれいなグラフィックスを楽しみましょうというだけのこと．そのために知っておいてほしいことのいくつかを，本節で紹介いたします．



コマンドを並べて

絵を描く？ 何も考えることないでしょ．線を引くなら LINE 命令，円を描くのは CIRCLE 命令，必要なだけいくらでも並べればいいわけでしょ――といういい分は確かに真実であります．それが証拠にこんな絵を作ることもできます．*Figure 15* を見てください．これは *List 5* のプログラムを実行した結果です．

このようなプログラムを作るには，まず方眼紙に下絵を描いて，必要な座標をひろっていくとよいでしょう．

座標をひろう際に，注意しなければならないのはだ円の半径の決め方です．だ円は，その長径で大きさを指定します．つまり横長のだ円ならば x 軸方向のドット数，縦長のだ円ならば y 軸方向のドット数で指定するのです．

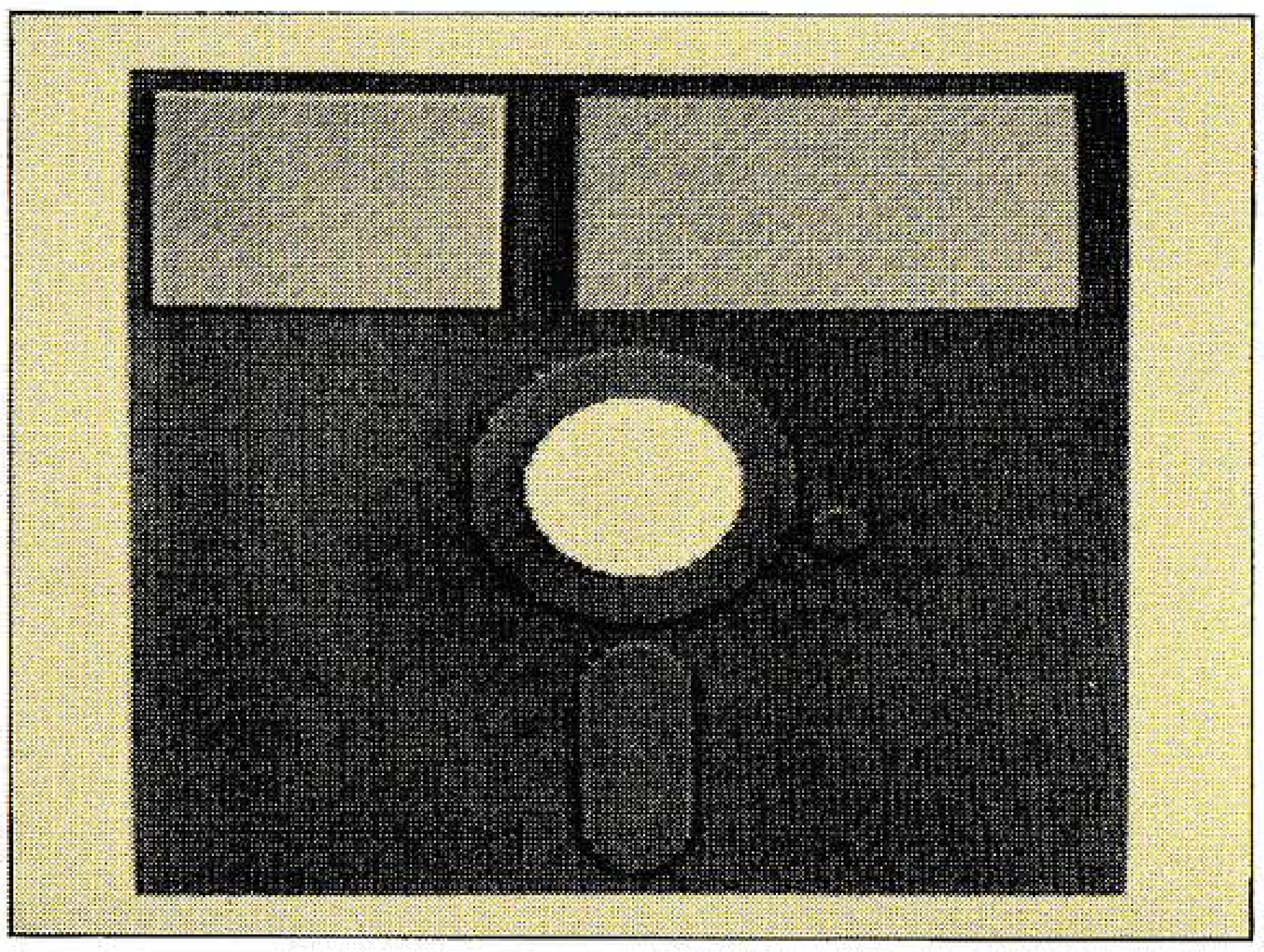


Figure 15

List 5

```
100 ' List   5
110 '
120 SCREEN 2 : COLOR 7,15,15 : CLS
130 PI=3.14159
140 LINE (35,10)-(225,190),4,BF
150 CIRCLE (130,100),30,13
160 PAINT (130,100),13
170 CIRCLE (130,100),19,15
180 PAINT (125,100),15
190 CIRCLE (168,110),5,13
200 PAINT (168,110),13
210 CIRCLE (130,145),10,13,0,PI
220 CIRCLE (130,175),10,13,PI,0
230 LINE (140,145)-(140,175),13
240 LINE (120,145)-(120,175),13
250 PAINT (130,150),13
260 LINE (40,15)-(105,60),11,BF
270 LINE (120,15)-(215,60),7,BF
280 I$=INPUT$(1):END
```

READ DATAの利用

コマンドを並べて絵を描くという前述の方法，何かもの足りないと思った人はいませんか．その人はなかなかスルドイ，将来有望です．いったいコマンド ズラズラ方式のどこがもの足りないのか，それは単純すぎるというところです．BASICの命令としてあらかじめ用意されている命令のひとつひとつを，そのまま使っているからです．

せめて，もうひと工夫，それが人間というものです．

そこで，絵を描くにはこういう方法もありますよ，という例を示します．*List 6* がそのプログラムです．210行以後は座標のデータ（2個1組でxとyを表す）の固まりで，これをREAD命令で読み出してきて直線を引きます（*Figure 16* 参照）．

160〜180行に注目．FOR〜NEXTループで，「あること」を23回繰り返しています．その「あること」というのは，READ命令でxとyの座標を読み出してきて現在の位置からその(x，y)に向けて直線を引く，

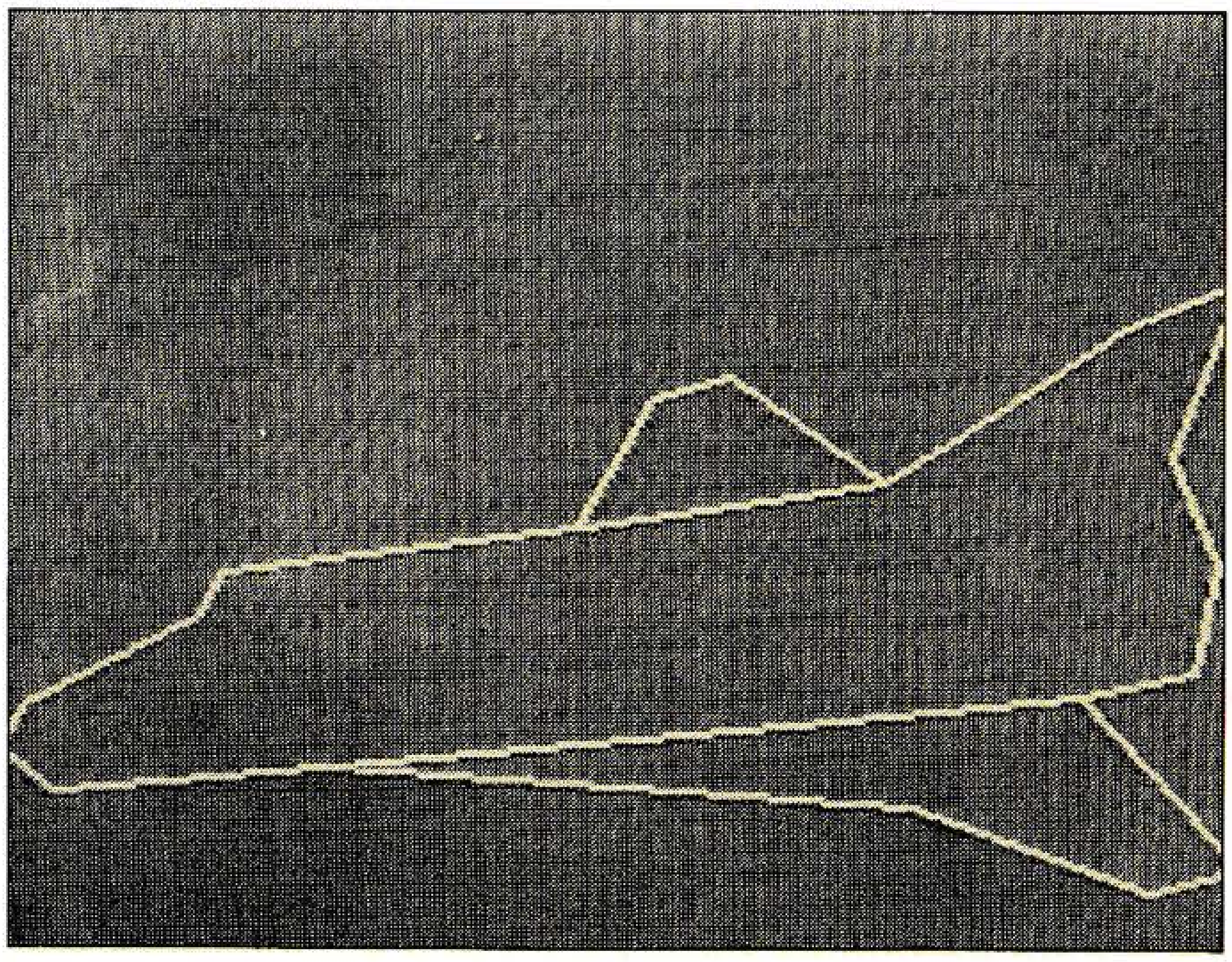


Figure 16

List 6

```
100 ' List   6
110 '
120 SCREEN 2 : COLOR 15,4,4 : CLS
130 RESTORE 210
140 C=15
150 READ X,Y : PSET (X,Y),C
160 FOR I=1 TO 23
170   READ X,Y : LINE -(X,Y),C
180 NEXT I
190 I$=INPUT$(1):END
200 '
210 DATA 180,95, 230,60, 255,50, 240,90, 250,115, 245,140
220 DATA 220,145, 255,185, 235,190, 185,170, 70,160, 220,145
230 DATA 70,160, 15,165, 5,155, 10,145, 45,125, 50,115
240 DATA 120,105, 125,95, 135,75, 150,70, 180,95, 120,105
```

という作業です．LINE命令で，「始点の座標を省略すると前回引いた直線の終点が今回の始点として使われる」という機能を，非常にうまく利用しています．こうしていわゆる一筆書きが描けるわけです．DATAを少し入れ換えただけで，簡単に図形を変更できます．そのときは，DATAの数によって160行のＩを変えなければなりません．

話が前後しますが，ここで150行を見てください．これは次の160行から始まる一筆書きに備えて，最初の点を決めています．この部分がないと，電源を入れた直後の「暗黙の始点」は（0，0）の位置になっているため画面左上から線が引かれてしまいますためしてごらんなさい．

角度と距離による位置の指定

自分で何から何までデザインしたグラフィックを画面に表示させてみるのも楽しみ方のひとつですが，コンピュータが独自に計算して作り出したグラフィックをボンヤリと眺めている，これもひまつぶしにはもってこいの遊びです．

もとは単純な図形でも(そう，たった1本の直線でも)，コンピュータの計算能力によって変化し，組み合わされていく様子は，その整然としたリズムの中に思いがけない美しさが発見できて楽しいものです．

このような計算によるグラフィックは次章でもいろいろと扱っていますが，中でひとつだけここで説明を加えておきたいこと

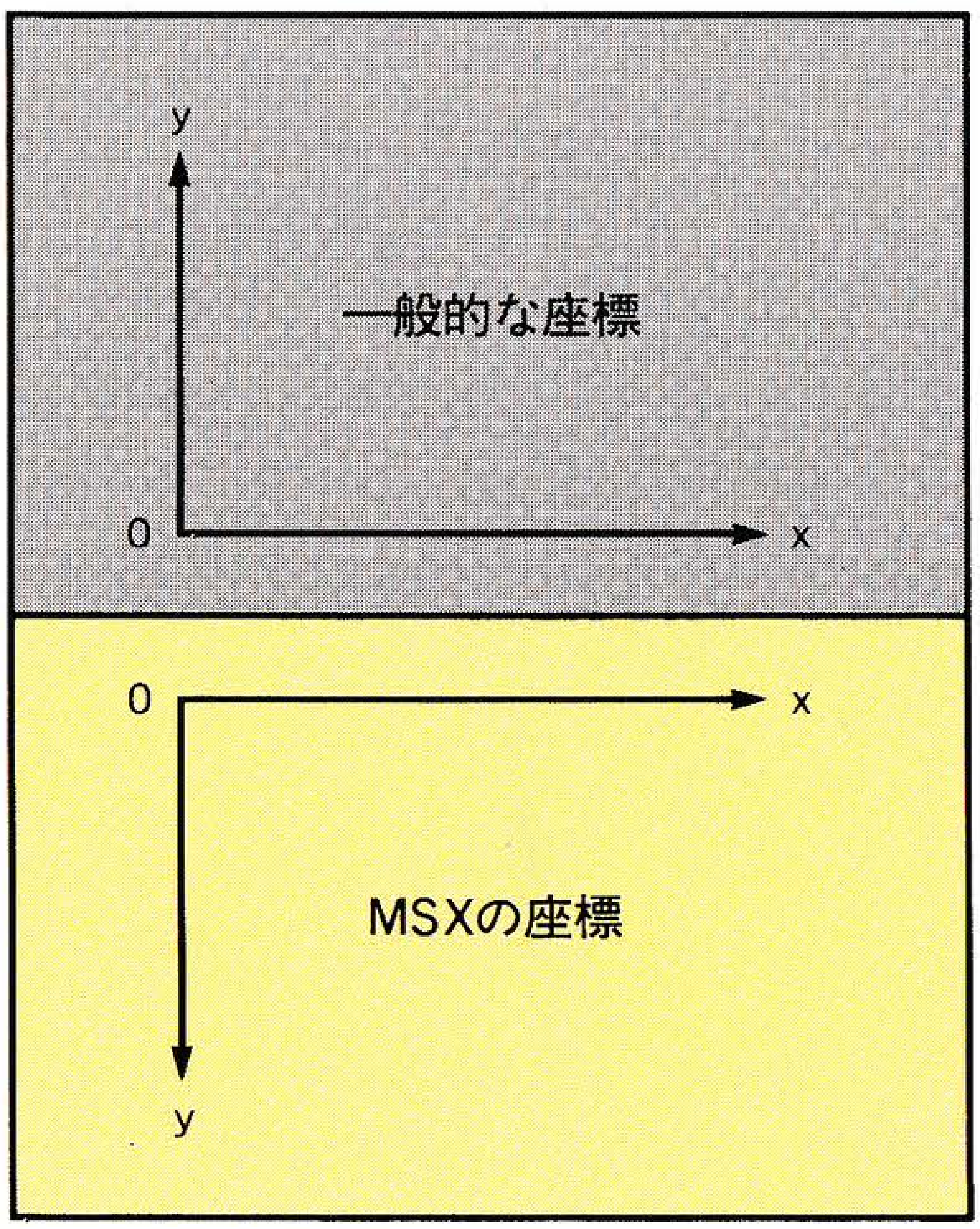


Figure 17

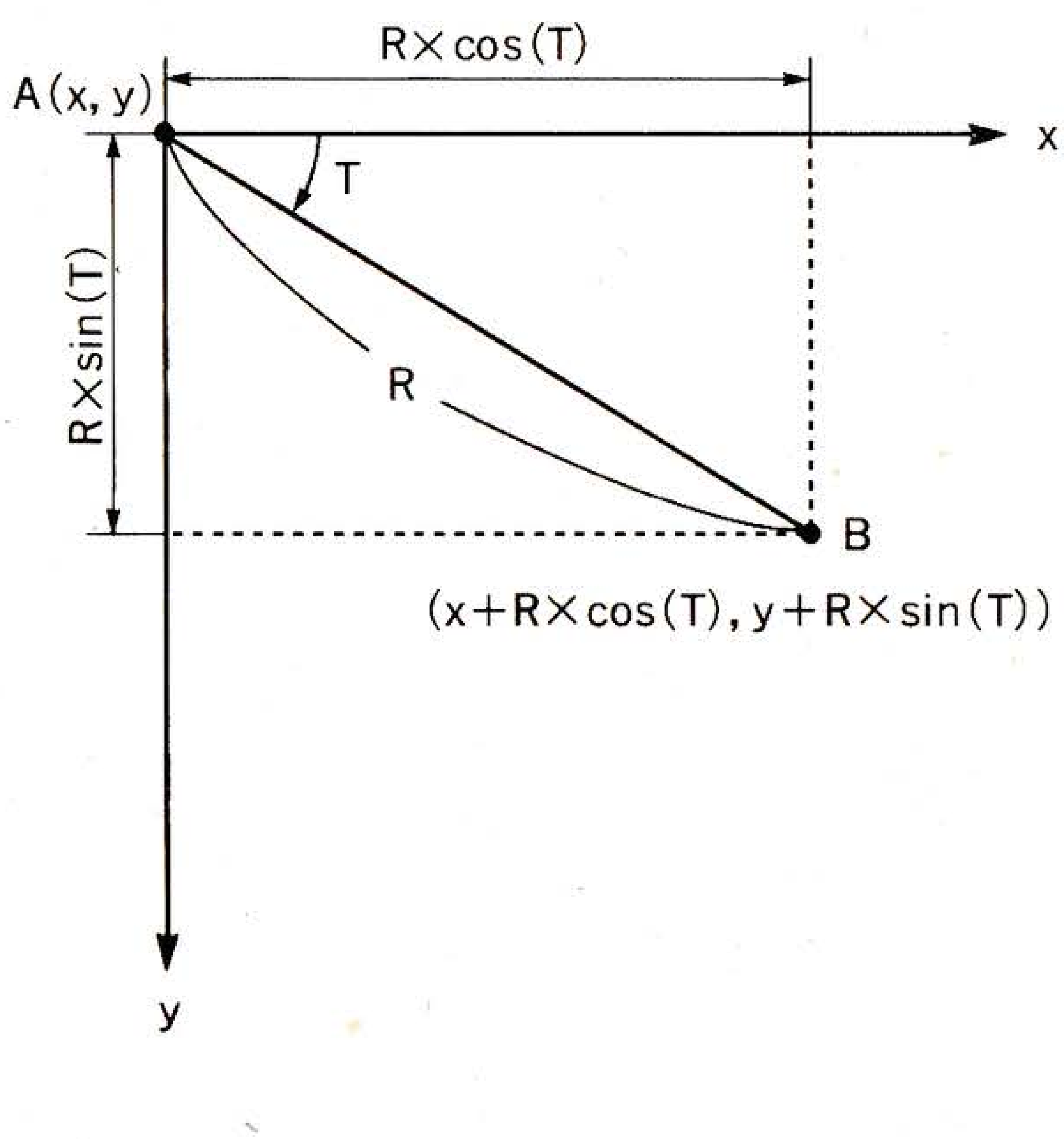


Figure 18

があります．sin 関数と cos 関数，例の三角関数というヤツの使い方です．

あれ，もう数学しないんじゃなかったの．やだよ，サイン，コサイン なんて……．

まあまあ，そんなこといわないで．むずかしいことはやりませんから．要するに高さを求めるのが sin 関数，横幅を求めるのが cos 関数ってこと．知ってりゃいいのはこの2つだけなんです．円を基本にしたグラフィックには，sin 関数と cos 関数が必要不可欠であるといいきって間違いなし，多少頭が痛くても，これを覚えておいて絶対にソンはありません．

MSX で使う座標は，x 座標が右方向を，y 座標が下方向を向いています．学校で習ったものと y 軸の向きが逆なので注意してください（*Figure 17* 参照）．

次に *Figure 18* を見てください．(x，y) にある点 A から直線距離で R，水平方向との角度が T だけ離れた点 B の位置を考えます．B までの横方向の距離すなわち x 座標の幅は R ×cos (T)，そして B までの縦方向の距離すなわち，y座標の幅はR×sin (T) として計算されます．よろしいですか，横が cos 関数，縦が sin 関数，くれぐれも間違えないように．

たとえば R を10cm に決めて角度 T を少しずつ変えながら B の位置に点を打っていけば，A を中心とした半径10cm の円が描けることになります．

もしもこの方法で，だ円を描こうとするのなら，縦横比を変えてやらなくてはなりせん．*List 7* を見てください．150，160 行で描いている円は計算された y 座標をそのまま使っているので円となりますが，170，180行の方は y 座標に P を掛けてだ円を

List 7

```
100 ' List . 7
110 '
120 SCREEN 2 : COLOR 15,13,13 : CLS
130 PI=3.14159 : P=.5
140 FOR T=0 TO 2*PI STEP PI/20
150   X=40*COS(T) : Y=40*SIN(T)
160   LINE (80,90)-(80+X,90+Y),15
170   X=40*COS(T) : Y=40*SIN(T)*P
180   LINE (180,90)-(180+X,90+Y),15
190 NEXT T
200 I$=INPUT$(1):END
```

List 7a

```
140 FOR D=0 TO 360 STEP 9 : T=D*PI/180
190 NEXT D
```

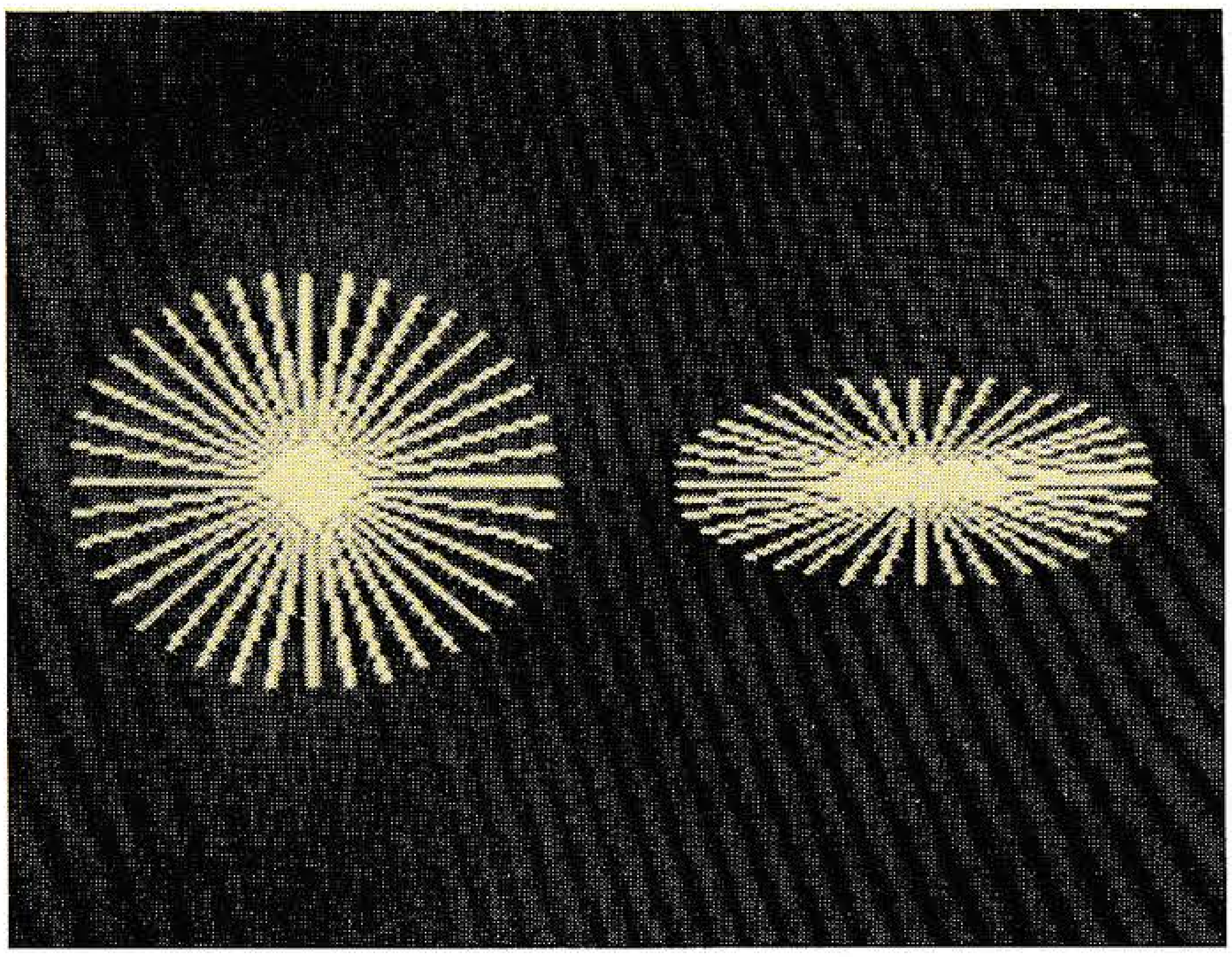


Figure 19

描いています（*Figure 19* 参照）.

三角関数で，もうひとついっておかなければならないのは角度 T の単位のことです．MSX-BASIC の三角関数は角度をラジアンという単位で指定します．この単位は円周360度を 2 π（≒6.28）ラジアンとして換算するもので，1 度がπ／180ラジアンに相当します．D 度なら D×（π／180）ラジアンです．しかし，これは数学的にはとても便利な単位ですが実用的には使いづらいものです．

このプログラムの中では T を0ラジアンから2πラジアンまで，直接にラジアン単位で指定していますが，もしラジアンがわか

List 8

```
100 ' List   8
110 '
120 SCREEN 2 : COLOR 15,1,1 : CLS
130 LINE (0,0)-(255,191),7,B
140 FOR I=0 TO 255 STEP 32
150   LINE (I,0)-(I,191),7
160 NEXT I
170 FOR I=0 TO 191 STEP 32
180   LINE (0,I)-(255,I),7
190 NEXT I
200 '
210 FOR L=1 TO 500
220   X=INT(RND(1)*256)
230   Y=INT(RND(1)*192)
240   PSET (X,Y),15
250 NEXT L
260 I$=INPUT$(1):END
```

List 8a

```
220   X=INT((RND(1)+RND(1)+RND(1))/3*256)
230   Y=INT((RND(1)+RND(1)+RND(1))/3*192)
```

らなければ，140，190行を *List 7a* のように変えると，おなじみの〝度〟を単位としてループを回すことができます．

乱数の利用

乱数は，コンピュータ・グラフィックスに欠かせないスパイスともいうべきものです．乱数を使用する目的は大まかに，次の2つが挙げられます．

まずひとつ目は，文字どおりランダムな効果を出すため，つまり，ここかと思えばまたあちら，グラフィックの現れる位置を予期できないおもしろさをねらうというものです．

また，乱数を使うもうひとつの目的は，グラフィックに自然の趣きを与えるということ．森の木々も海辺の砂粒も，遠くから眺めれば皆一様に見えても，実際にはそれぞれが個性を持った形の集まりです．そのような状況を画面上に表現する場合には，コンピュータの規則正しい計算力はかえって不都合，乱数によって得られる散らばりが自然の味を創り出すのです．

List 8 は，後者の例といってよいでしょう．恒星間宇宙船（スターシップ）の大映像盤（ディスプレイスクリーン）に広がる宇宙の星々の雰囲気が出ていませんか，これ（*Figure 20* 参照）．

プログラム中で乱数を使用するためには，乱数関数 RND(1) を用いるということはもう知っていますよね？ この RND(1)は，0以上1未満の範囲の乱数を得るもので，用途に応じてこれにいろいろと手を加えて，望みの範囲の乱数を得るわけです．

ここに挙げたプログラムの220行と230行では，x 座標を0～255の整数乱数，y 座標を0～191の整数乱数で得ています．このように，0から N−1までの整数を値とする乱数が欲しい場合には，次の式を用います．

```
INT(RND(1)*N) ························(a)
```

0から N−1までのN個(注意して！N−1個ではありません）の整数のうちのひとつが，この式の値となります．そして，この式を基本として，次のような応用もできます．

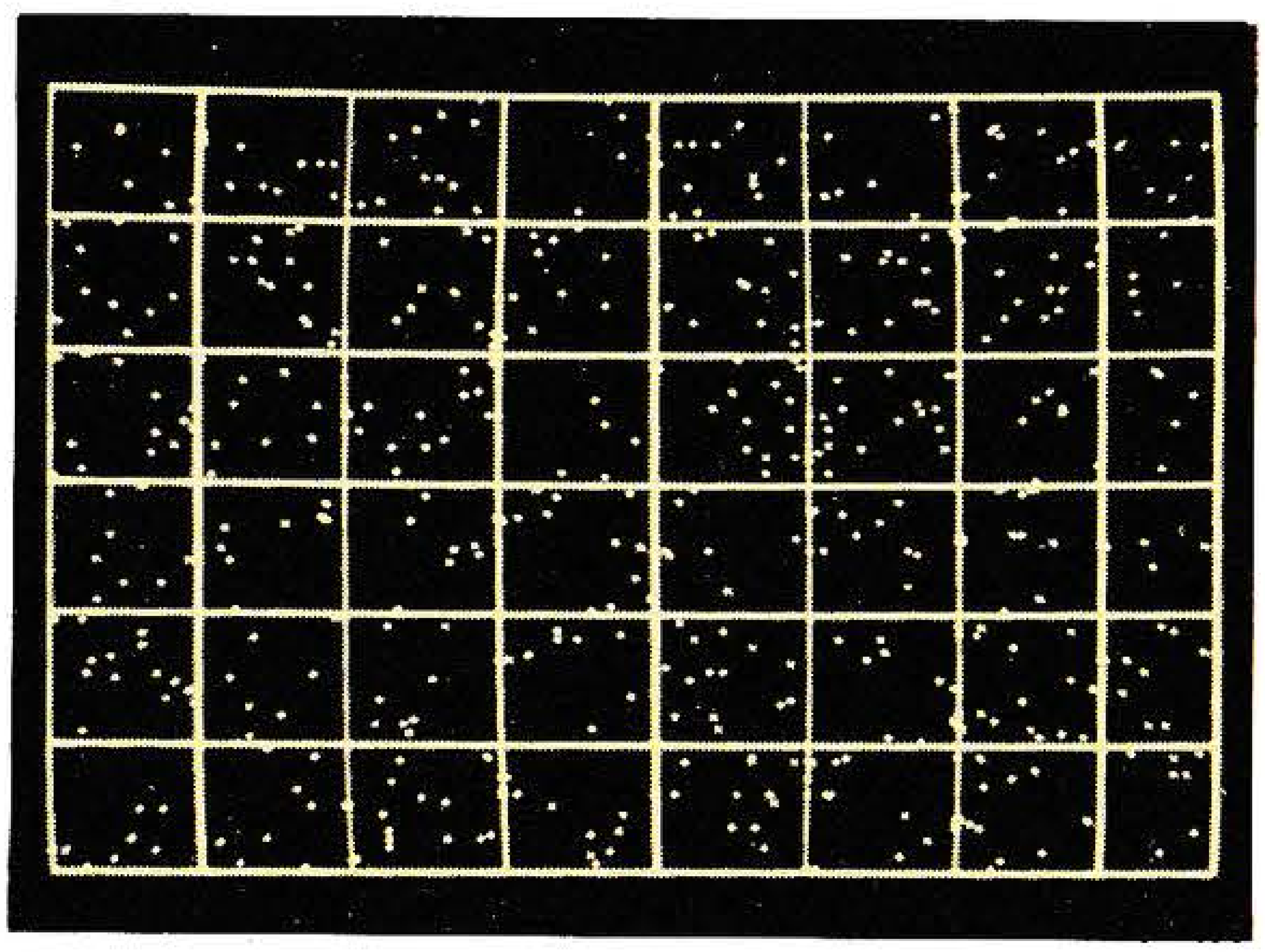
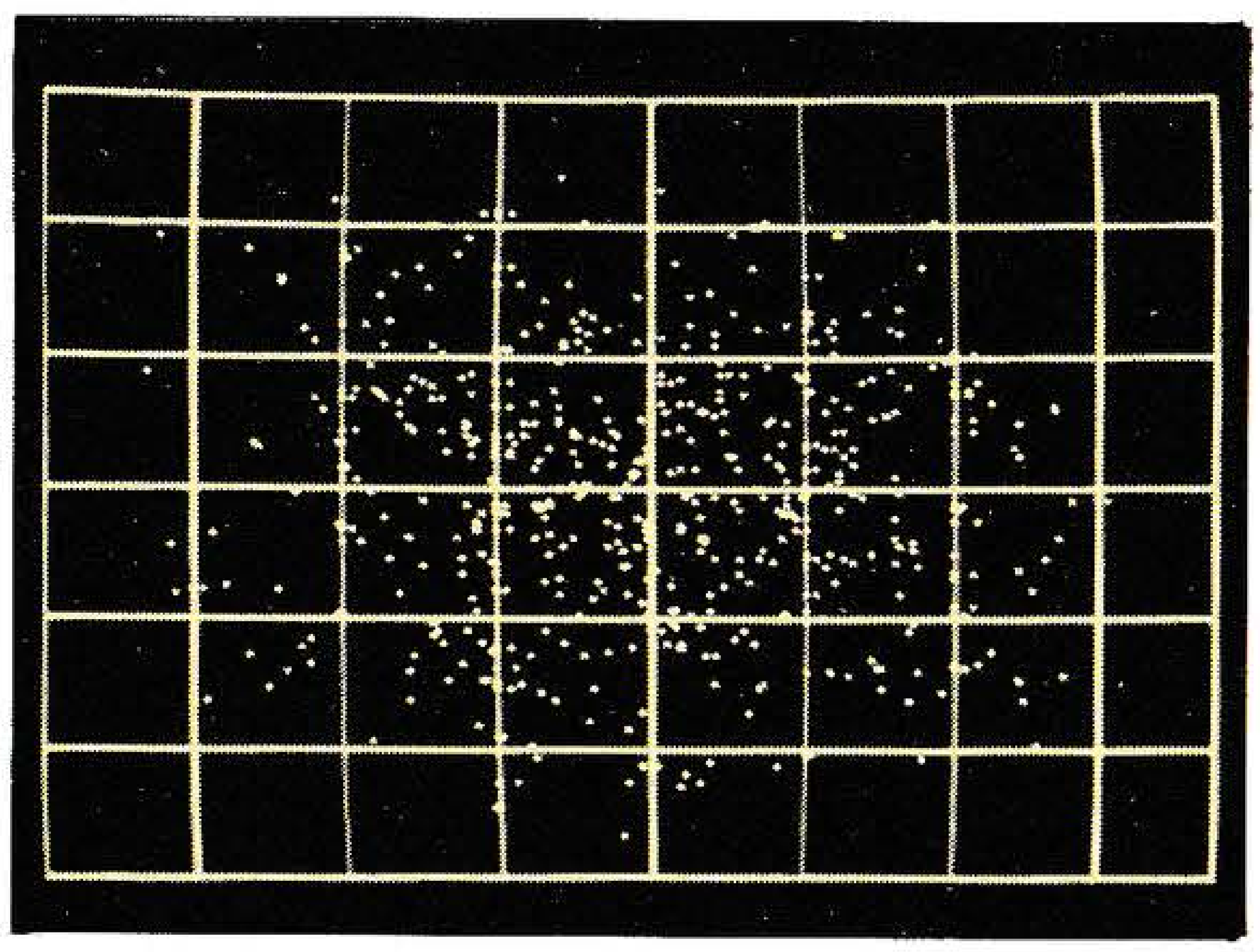


Figure 20 (List 8の実行結果)



(List 8aの実行結果)

● 1から6までの整数をランダムに出す

```
INT(RND(1)*6)+1
```

● 40, 50, 60の3つの数のうち1個をランダムに出す

```
INT(RND(1)*3)*10+40
```

● −2から＋2までの整数をランダムに出す

```
INT(RND(1)*5)-2
```

などなど．以下次章でもこの乱数の式は数値の範囲をいろいろ変えて登場します．

ところで(a)式の乱数は，どの値の現れる確率も N 回に1回で等しくなっています．これを，ある特定の数値がよく現れるように手を加える方法もあります．その簡単な例を2つ紹介しましょう．

1つ目は，ある範囲の中の中心部がよく出るようにする方法．これは RND (1) を N 個足します．すると0以上 N 未満の範囲の乱数が得られますが，出現確率は N／2が最も高くて，両端の0や N に近い値ほど低くなります．ちょうど西洋の鐘（ベル）のような形の確率分布になります．N を大きくするほど，つまり RND (1) をたくさん足すほど，中心部の確率が高く，両端は低くなります（*Figure 21* 参照）．

List 8 の220，230行を *List 8a* のように変えてみてください．星が画面の中心部に集まってきます．RND (1)を，3個でなく4個，5個とたくさん足すに従って，星が集まる度合も大きくなります．

乱数の確率分布を変える2つ目の方法は，RND (1) を掛け合わせるというものです．この場合は，乱数の範囲は0以上1未満で変化しませんが，0に近い部分の出現確率が高く，1の近くはほとんど出ません．つまり，これは *Figure 22* のような形になります．

List 8a の（RND (1)+RND (1)+RND (1))／3 の代わりに，RND (1) * RND (1) を用いると星は画面の左上に集まってくるでしょう．

乱数は，何の用意もなしに使うと，プログラムを走らせるたびにまったく同じ順序で値が現れます．たとえば，

```
10 PRINT RND(1);RND(1);
   RND(1)
```

というプログラムを実行してみてください．

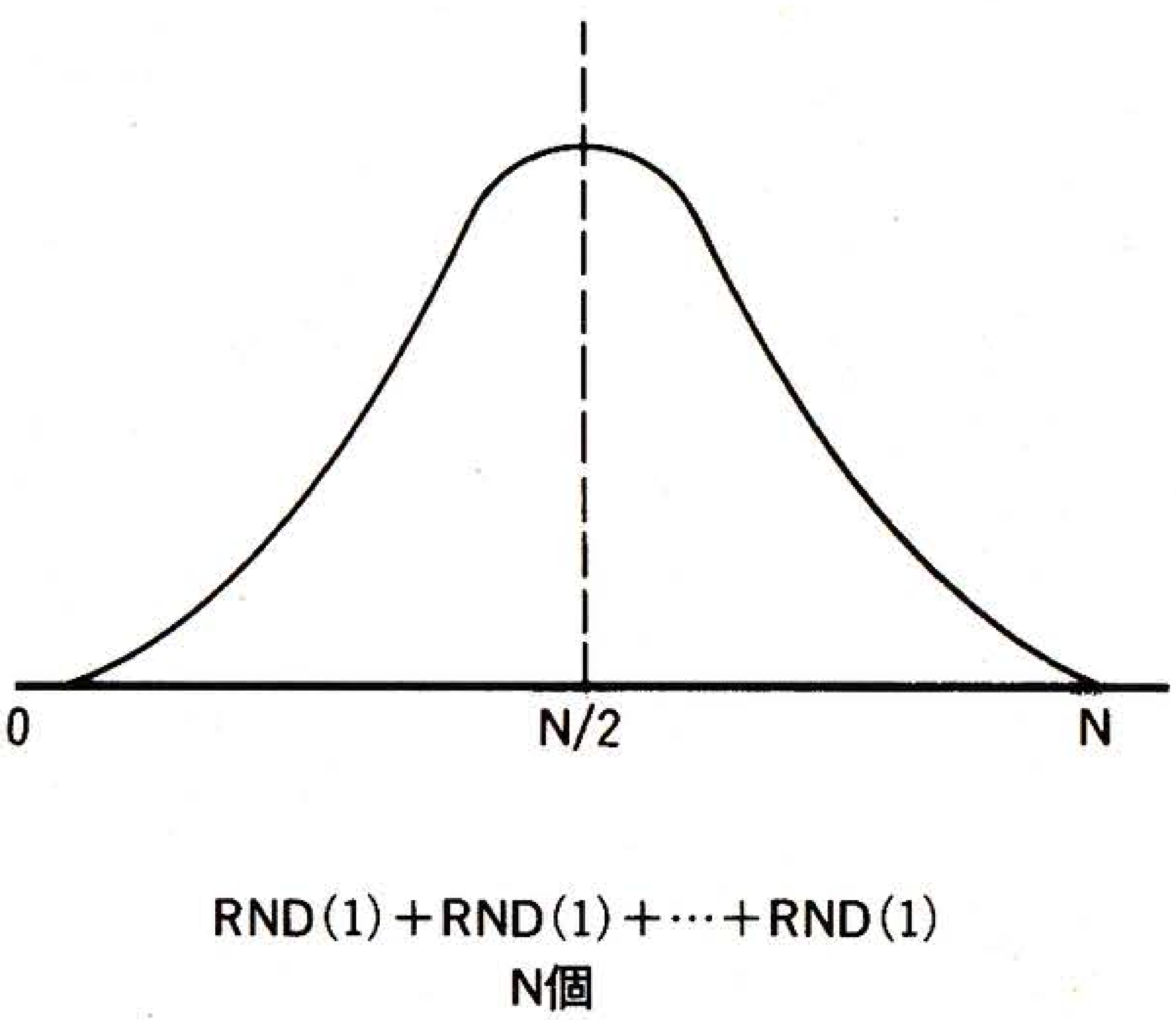


Figure 21

かならず，

0.59521943994623

0.10658628050158

0.76597651772823

と3個の数値が表示されるはずです．これでは，ゲームなどに乱数を使う場合，毎回同じ局面が現れてきて興ざめということになります．

これを防ぐためには RND (1) の代わりに RND(−TIME)を用います．*List 9* を実行してください．実行するたびに異なるグラフが現れます．

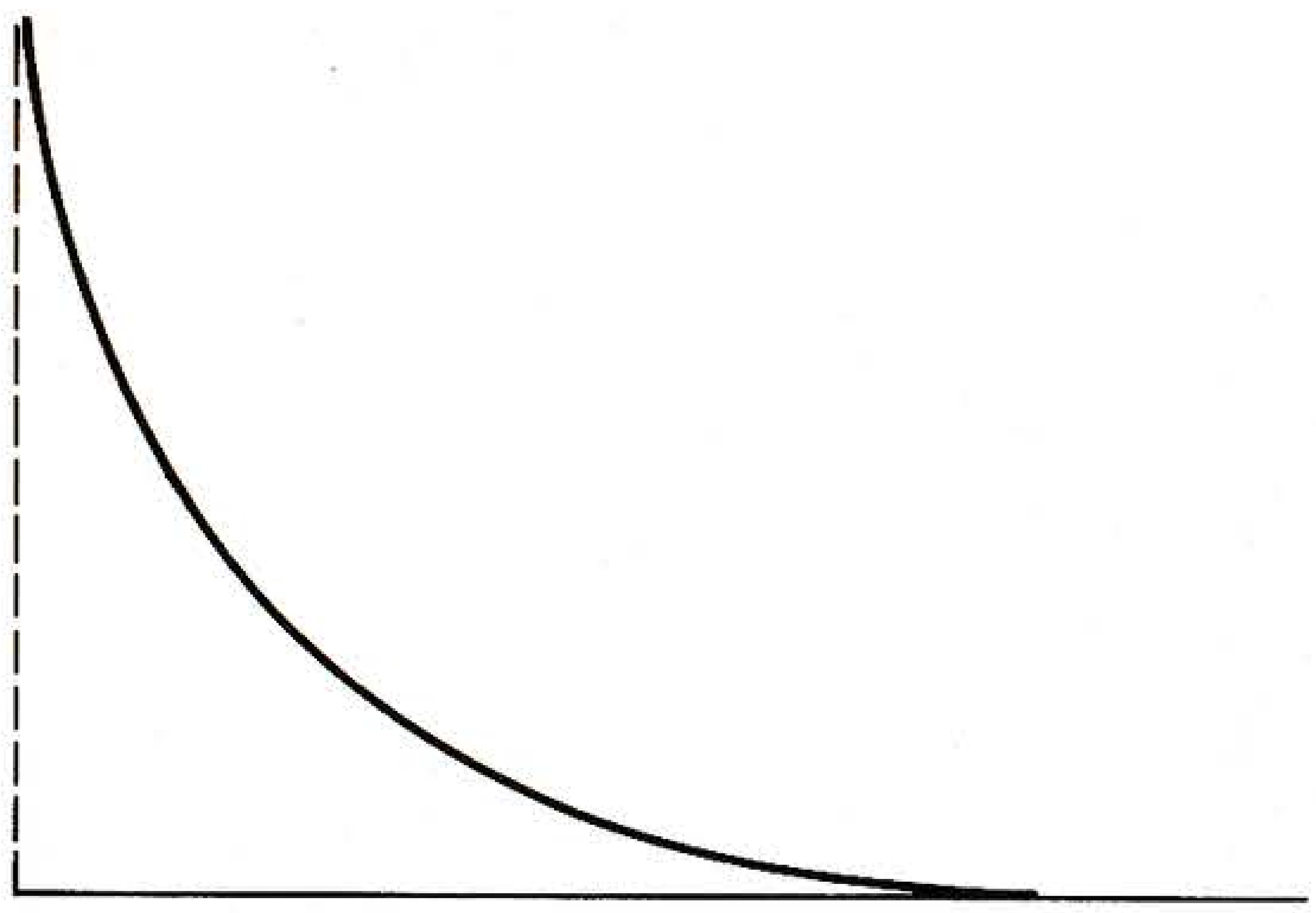


Figure 22

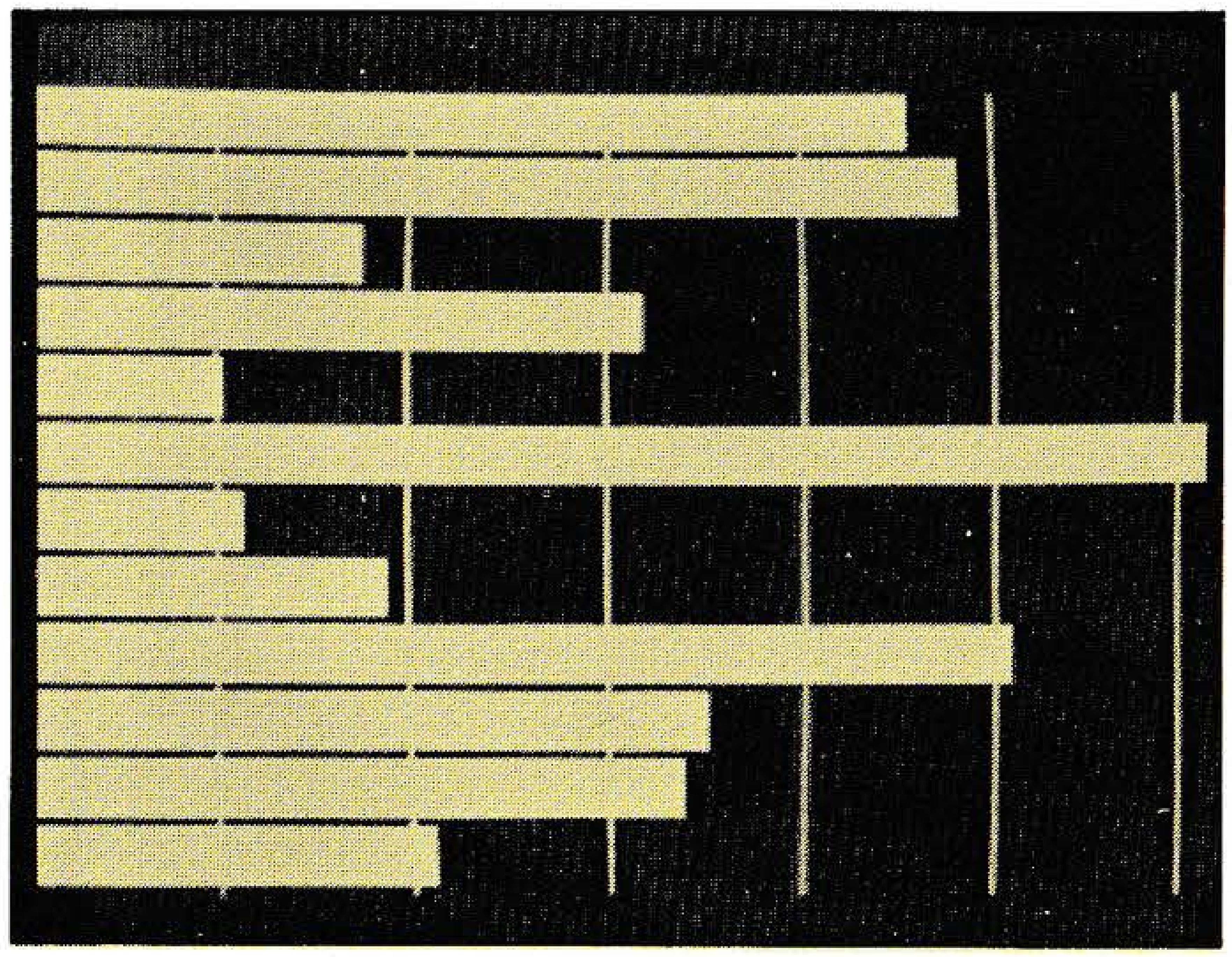


Figure 23

List 9

```
100 ' List   9
110 '
120 SCREEN 2 : COLOR 15,4,4 : CLS
130 DM=RND(-TIME)
140 '
150 FOR X=0 TO 255 STEP 40
160   LINE (X,0)-(X,191),15
170 NEXT X
180 '
190 FOR Y=0 TO 191 STEP 16
200   X=INT(RND(1)*255)
210   LINE (0,Y)-(X,Y+13),15,BF
220 NEXT Y
230 I$=INPUT$(1):END
```

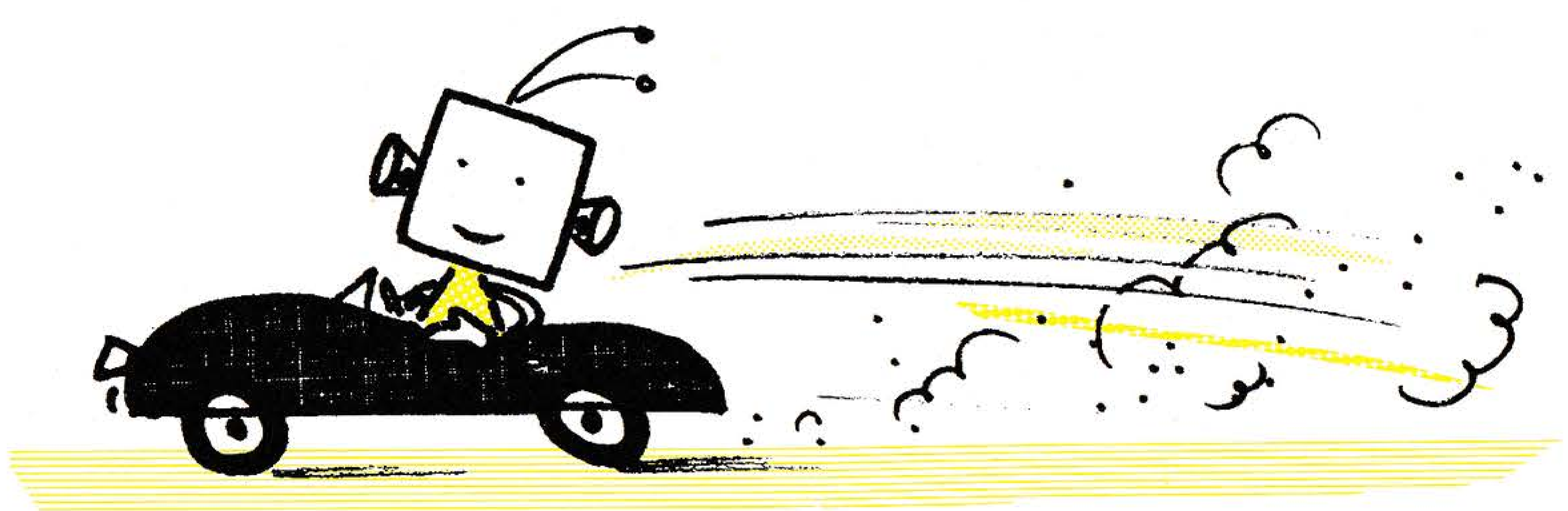
動くグラフィックのこころみ

2月30日午後10時，指定の場所に来られたし…….差出人不明の手紙を受け取った私は，好奇心も手伝って，よせばいいのにノコノコと出かけて行った．ドアを開けるとむやみに長いだけの廊下が奥へと通じていた．コツ，コツ，私の靴音のみが響く．廊下の中ほどの壁に一枚の絵がかかっていた．

おやおや，ジョコンダのおばさんか．モナ・リザといえば通りはよいが．何たる俗趣味だ．そう思いながら歩みを進めたとき，私の視線の端で何か動いたような気がした．はてな，今あの絵の方から何かの気配を感じたが？ 振り向くと絵の中の彼女が私にニヤリと笑いかけた…….

と，やや不気味な雰囲気で本節を始めてみましたが，これは別にこれから説明することと大した関係はありません．ただ，私たち人間は動くものに対して非常に敏感だ，ということをいってみたかっただけなのです．考えごとをしていて目の前の電柱に気づかないことはあっても，その視界の中に犬なり猫なりがとびこんできたら，驚かずにはいられません．いや，これは人間だけではありません．カエルは動くものしかエサだとは思わないといわれますし，卵からかえったヒヨコは，初めて見た動く物体を母親だと思い込むそうです．動きのあるものに興味を覚えるのは，生まれたときから備わった動物の本能というものなのでしょう．

グラフィックに動きを与えたとき，それが静止している画面の何倍もおもしろく感じられるのは，そういったことと関係があるかもしれませんね．そこで本節では，グラフィックに動きを与える，というテーマにそって説明していきます．



同じ形を並べて

ここに1匹のネズミがいます．チューと鳴いて1匹増えました．もう一度チューと鳴いてさらに1匹増えました．横一列に並んで，はいおしまい．

なんだ，これは数が増えているだけで，ネズミはちっとも動いていないじゃないかって？ いやまだ始まったばかりなんですよ．そんなに早く目的を達成してはつまらないでしょう．こういう地道な努力の積み重ねが大切なんですよ，ホントに（*Figure 24* 参照）．

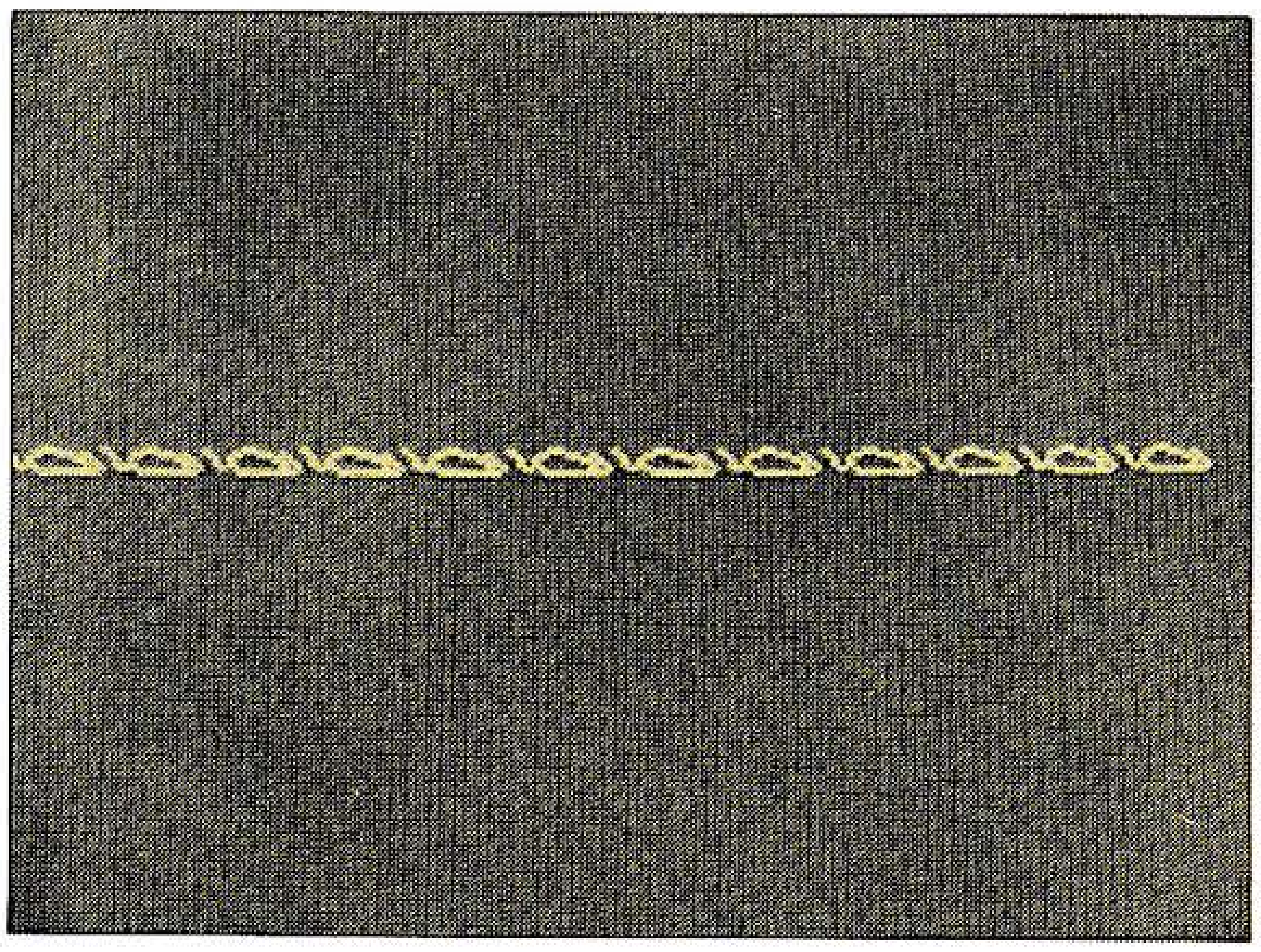


Figure 24

さて，このプログラムはどのように作られているのでしょうか．*List 10* を見てください．190行から，1匹分のネズミを描くサブルーチンです．

この中の座標はすべて，(x，y)の位置を基準として，右に何ドット左に何ドットという形で表され，基準となる（x，y）の値を変化させれば，画面のどの位置にでもネズミが表示できるようになっています．位置をある一か所に固定してしまうのではなく，基準となる座標からの相対的な変位として与えるこのような方式を，相対座標による位置指定といいます．

ネズミが横一列に並ぶのは，130行で y 座

List 10

```
100 ' List   10
110 '
120 SCREEN 2 : COLOR 15,4,4 : CLS
130 Y=90
140 FOR X=0 TO 230 STEP 20
150   GOSUB 190 : PLAY "T128O6C64"
160 NEXT X
170 I$=INPUT$(1):END
180 '
190 DRAW "S4A0 BM=X;,=Y; BM+3,+0"
200 DRAW "C15 M+3,+4 M+2,-1 M+1,-2"
210 DRAW "M+3,-1 M+6,+2 M+1,-1"
220 DRAW "M+1,+1 M+2,+2 M-1,+1"
230 DRAW "M-11,+0 M-2,-2 BM+11,+0"
240 DRAW "M+0,+1"
250 RETURN
```

標を固定し，140〜160行のループでｘ座標を次々と変化させているからです．逆に，ｘ座標を一定にしてｙ座標を変化させれば，ネズミは縦に積み重ねられます．そしてｘとｙの両座標を同時に変化させていけば，その変化の方法によって，円形や斜めにも並ばせることもできます．

表示位置が斜めの直線に沿って移動するグラフィックの例を *List 11* に示します．このプログラムでは，変数Ｉの変化に伴い，ｘ座標は XS ずつ減っていき，ｙ座標は YS ずつ増えていきます．その両者を組み合わせた座標の位置が，ちょうど坂をころがる円の中心となっていくのです（*Figure 25* 参照）．

なお，160行の L の値を変えると，画面に表示される円の個数が変わるように，プログラムを作ってみました．これはほんのおまけです．

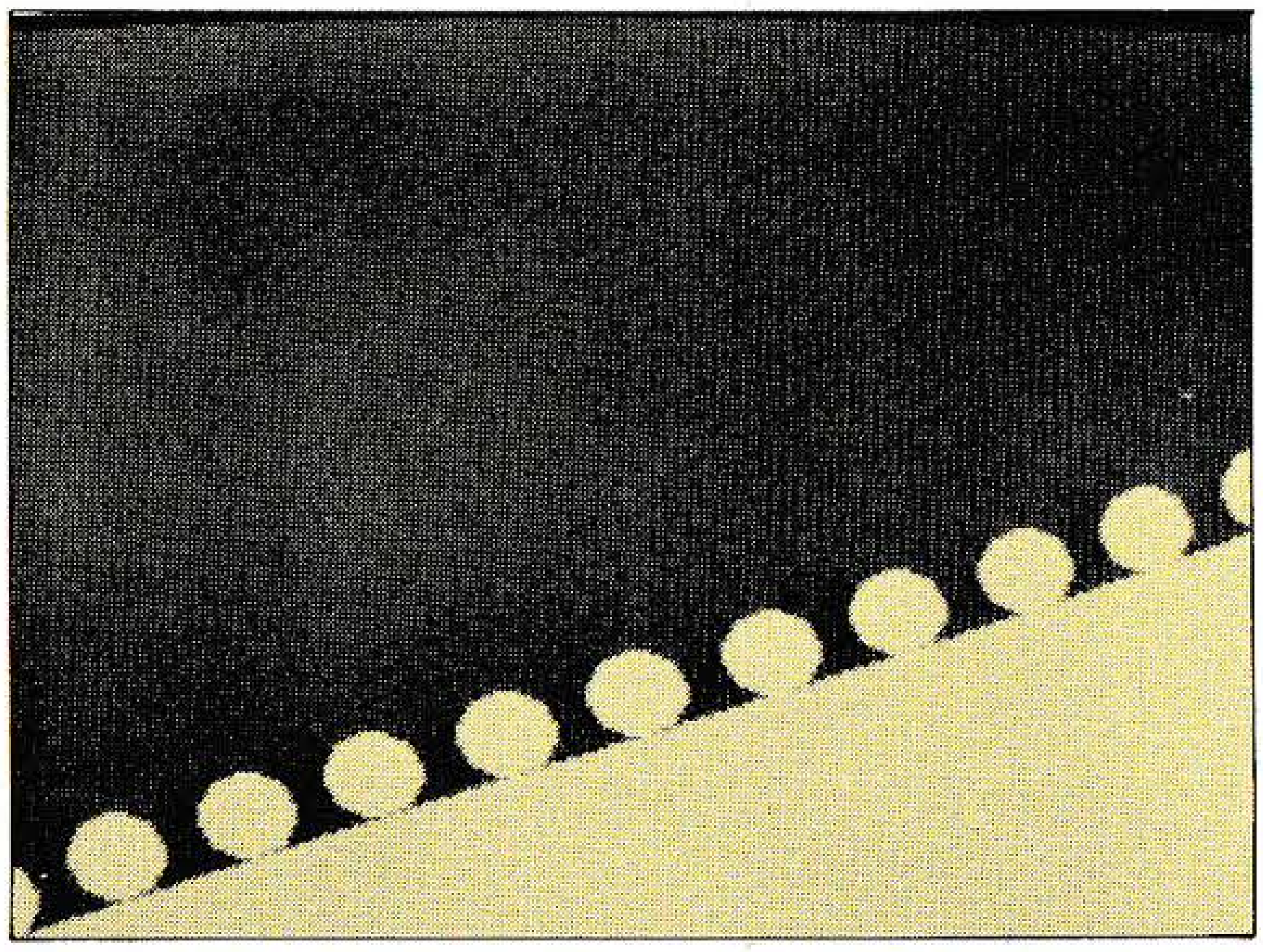


Figure 25

周期的繰返し

位置の移動だけがグラフィックに与える「動き」ではありません．さてこんなコミカルな信号機はいかが（*Figure 26* 参照）．

List 12 のプログラムを実行すると，2つ目玉の赤信号が点滅を始めます．この動

List 11

```
100 ' List   11
110 '
120 SCREEN 2 : COLOR 15,13,13 : CLS
130 DRAW "S4A0 C15 BM0,191 M255,191 M255,100 M0,191"
140 PAINT (250,190),15
150 '
160 L=10
170 XS=255/L : YS=91/L
180 FOR I=0 TO L
190   X=255-XS*I
200   Y=90+YS*I
210   CIRCLE(X,Y),9,15
220   PAINT(X,Y),15
230   PLAY "T128L64O2A"
240 NEXT I
250 I$=INPUT$(1):END
```

作を行わせているのが，220行から290行にかけてのループ部分，その中で重要な働きをしているのが F という変数です．

Fには，最初210行で1という値を与えておきます．間の数行分の説明はちょっと後回しにして，280行を見てください．F の値はここで変化します．今，1であった F の値は，この行の命令を実行すると0になります．ループによって，再びこの280行の命令が実行されると，F はまた1に戻ります．結局ループを回るたびに，F の値は1と0を交互に変化するのです．

このFの値によって，どちらの目玉を点灯するのか決めているのが230行と240行．しばらく時間をおいたあと，260，270行で黒く塗りつぶします．

List 12

```
100 ' List   12
110 '
120 SCREEN 2 : COLOR 15,2,2 : CLS
130 LINE (120,70)-(168,90),15,BF
140 DRAW "S4A0 BM120,90 C15"
150 DRAW "M+0,+56 M+4,+3 M-12,+0"
160 DRAW "M+4,-3 M+0,-80 M-2,-2"
170 DRAW "M+8,+0 M-2,+2 M+0,+4"
180 CIRCLE (134,80),6,1 : PAINT (134,80),1
190 CIRCLE (154,80),6,1 : PAINT (154,80),1
200 '
210 F=1
220 PLAY "T255L6404A"
230   IF F=1 THEN CIRCLE(134,80),6,5 : PAINT(134,80),5
240   IF F=0 THEN CIRCLE(154,80),6,9 : PAINT(154,80),9
250   FOR T=1 TO 200 : NEXT T
260   IF F=1 THEN CIRCLE(134,80),6,1 : PAINT(134,80),1
270   IF F=0 THEN CIRCLE(154,80),6,1 : PAINT(154,80),1
280   F=1-F
290 GOTO 220
```



List 12a

```
210 X=134
220 PLAY "T255L6404A"
230   CIRCLE(X,80),6,13 : PAINT(X,80),13
240   FOR T=1 TO 200 : NEXT T
250   CIRCLE(X,80),6,1  : PAINT(X,80),1
260   X=(134+154)-X
270 GOTO 220
```

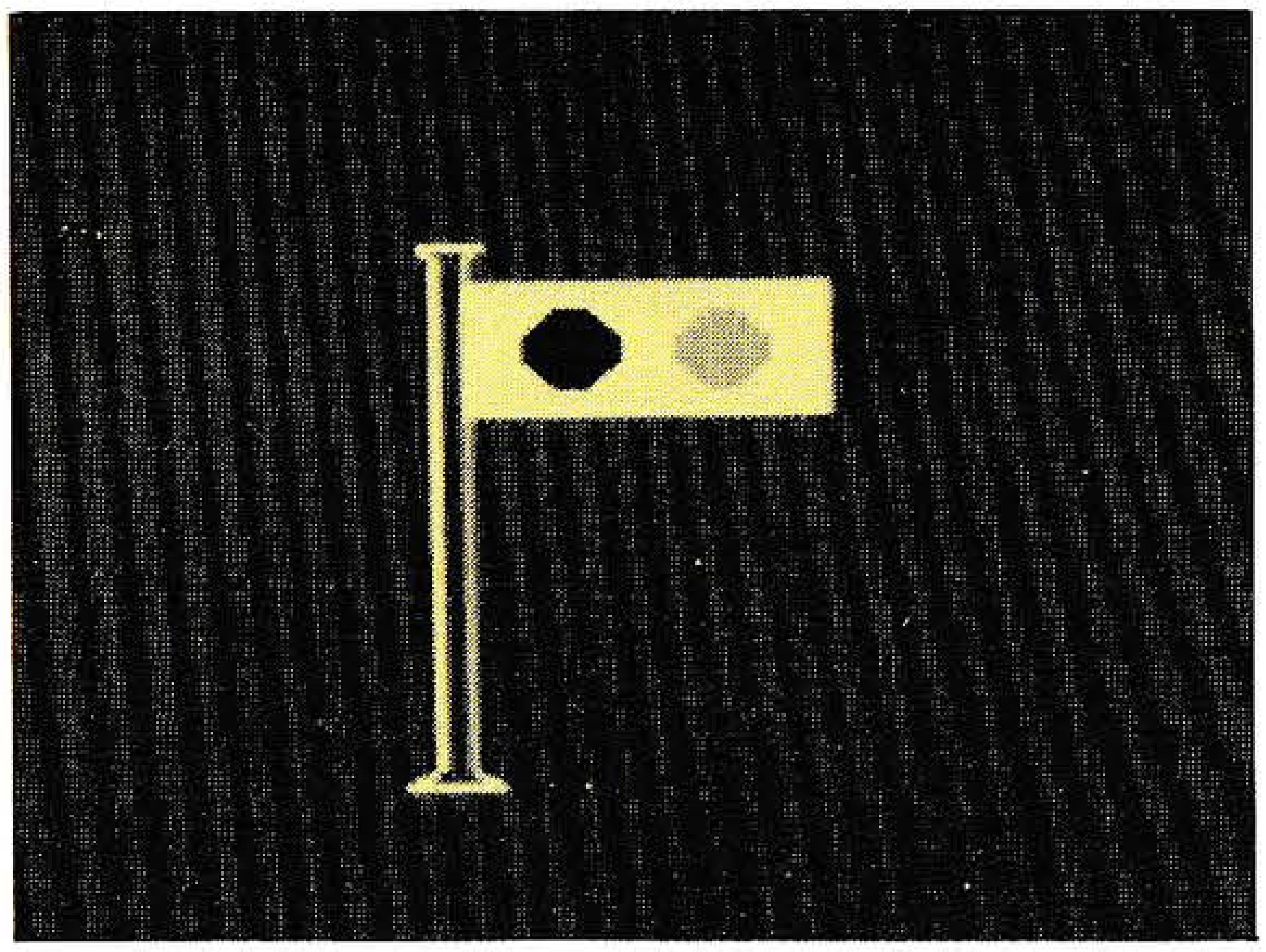


Figure 26

このプログラムでは，変数に1と0とを繰り返させましたが，他のどんな数を繰り返させることもできます．変数 X に M と N の2数を繰り返させる式は，次のとおりです．

X＝(M＋N)－X

はじめ X に M という値を代入しておけば，この代入文を1回実行するごとに，X の値は M から N へ，N から M へと交互に変化します．たとえば，このプログラムの210行以降の行を *List 12a* のように変えてみましょう．これでも結果は同じです．

この場合は，点灯する目玉の x 座標，134と154を直接に繰り返させているのです．

移動するグラフィック

では今度は，グラフィックが本当に位置を移していくプログラムを作ってみましょう．これは何のことはない簡単な原理でして，基本的には *List 10* のネズミと同様，次々と位置を変えながら絵を描いていけばよいのです．ただし，新しい位置に絵を出す前に，古い位置の絵を消すこと．そうすれば，その絵はどんどん先へ進んでいくように見えます．

List 13 のプログラムの，240行以降のサブルーチンが，走っている人間の絵を描きます．やはり，相対座標形式で，どこに描くかを指定できるようにしています．また変数 C の値によって，ランナーの色を変えることもできます．

160～210行が，ランナーを移動させるループです．2人のランナーを表示するのは170行と180行，上のランナーを早く先へ進ませるため，170行のランナーの x 座標は180行のランナーよりも大きな値になるようになっています．

この2人を消すために，190行と200行が実行されます．ランナーを描くサブルーチンを調べてみると，ランナーの絵を構成するドットの中で，最も左の x 座標は X＋2，最も右の x 座標は X＋17，そして y 座標は上の端が Y＋2，下の端が Y＋26，この範囲内をすべて塗りつぶしてしまえば，(X，Y) の位置にいるランナー1人を消すことができるのです．(*Figure 27* 参照)．

この *List 13* は，現在表示している位置の絵を消してから，次の新しい位置に絵を出すという順序での移動の様子を示しました．この方法ですと一瞬，画面上からすべての絵が消えている時間があることになります．これではサビシイ！というので，次に，画面上には常に少なくとも1個の絵が表示されている方法を見てみましょう．

List 14 のプログラムでは，220行から

のサブルーチンが，画面上(X，Y)の位置に1台の車を描きます．これをどのように動かしていくのでしょうか？

130行で決めている（X0，Y0）は，現在の車の位置です．140行で FOR～NEXTのループに入りますが，150行で計算している(X1，Y1)は，車が次に位置する座標です．さてそこで，160行でまず次の位置に車を表示してから，170行で今いる場所の車を消します．そして180行では，さきほど計算した新しい位置の座標が，新しい「現在」の座標として受け渡されます．あとは同じことを150行から繰り返し，こうして車は移動していきます（*Figure 28* 参照）．

たいへんやっかいな作業ですが，こうすれば画面上から絵が消える瞬間はまったくありません．以上の理屈，のみこめましたか？

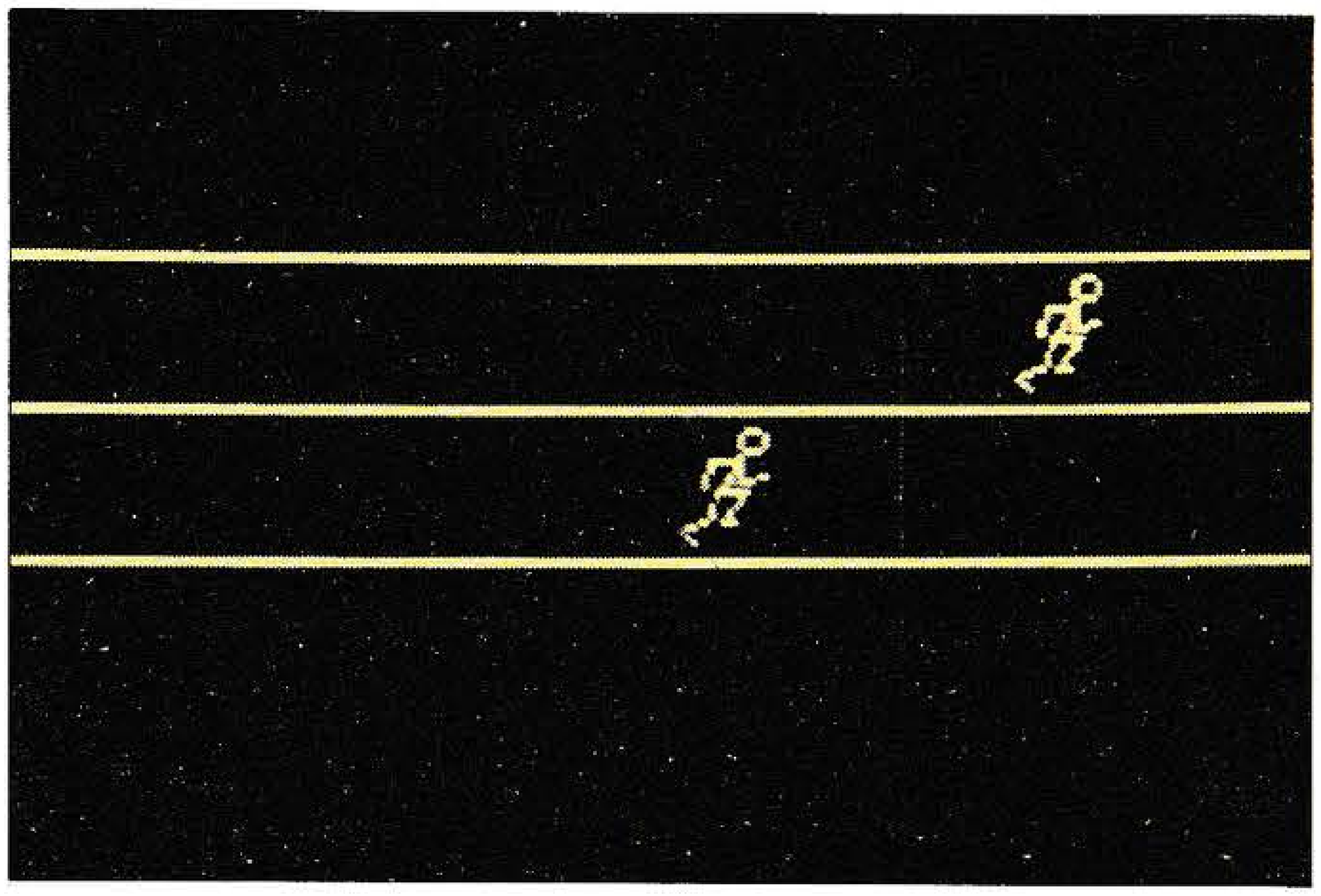


Figure 27

List 13

```
100 ' List   13
110 '
120 SCREEN 2 : COLOR 15,1,1 : CLS
130 FOR Y=50 TO 110 STEP 30
140   LINE (0,Y)-(255,Y),15
150 NEXT Y
160 FOR I=0 TO 22
170   X=I*10 : Y=50 : C=7  : GOSUB 240
180   X=I*7  : Y=80 : C=13 : GOSUB 240
190   X=I*10 : Y=50 : LINE (X+2,Y+2)-(X+17,Y+26),1,BF
200   X=I*7  : Y=80 : LINE (X+2,Y+2)-(X+17,Y+26),1,BF
210 NEXT I
220 END
230 '
240 CIRCLE(X+14,Y+6),2,C
250 LINE (X+13,Y+8)-(X+9,Y+16),C
260 DRAW "S4A0 BM=X;,=Y; BM+6,+15"
270 DRAW "C=C; M+1,-5 M+5,+0 M+0,+4"
280 DRAW "M+4,-1 BM-8,+3"
290 DRAW "M-1,+5 M-5,+3 M+2,+2"
300 DRAW "BM+4,-10 M+5,+0 M-4,+6"
310 DRAW "M+2,+0"
320 RETURN
```



Figure 28

キー入力による処理

List 15 を実行すると，画面の中央に水色の小さな四角形が現れます．a，c，x，zのキーを押すと，四角形が動いて跡を残していきます．またスペース バーを押すと，四角形の色が変わります（*Figure 29* 参照）．

これまで扱ってきたグラフィックは，その形がプログラム中ですべて決定され，人間の介する余地はありませんでした．しかし！ここでついに私たちは自分の手で，グラフィックを意のままに操ることができるようになったのです（？）．

List 14

```
100 ' List   14
110 '
120 SCREEN 2 : COLOR 15,4,4 : CLS
130 X0=230 : Y0=40
140 FOR I=1 TO 10
150   X1=X0-22 : Y1=Y0+10
160   C=15 : X=X1 : Y=Y1 : GOSUB 220
170   C=4  : X=X0 : Y=Y0 : GOSUB 220
180   X0=X1 : Y0=Y1
190 NEXT I
200 END
210 '
220 LINE (X+20,Y+9)-(X+10,Y+9),C
230 DRAW "S4A0 BM=X;,=Y; BM+5,+8 C=C;"
240 DRAW "M-5,-1 M+8,-4 M+4,-3"
250 DRAW "M+11,0 M+6,+5 M-1,+3 M-2,+0"
260 CIRCLE(X+8,Y+8),3,C
270 CIRCLE(X+23,Y+8),3,C
280 DRAW "BM=X;,=Y; BM+15,+1"
290 DRAW "M+5,+0 M+0,+3 M-9,+0"
300 DRAW "M+4,-3 M+7,+0 M+0,+3"
310 DRAW "M+4,+0 M-4,-3"
320 RETURN
```

とにかく，キー入力を判断してそれに応じた処理を行うのは，ゲームでも実用でもプログラム中にはかならず含まれる部分ですから，この手法を知っておくに越したことはありません．

このプログラムでは，X と Y で四角形を表示する位置を決めています．このプログラムでは，マルチカラーモードを利用しているので，PSET 命令でブロックを置くことになります．そこで，四角形を横に移動させるには X の値をプラスまたはマイナス4，縦に動かすには Y の値をプラスまた

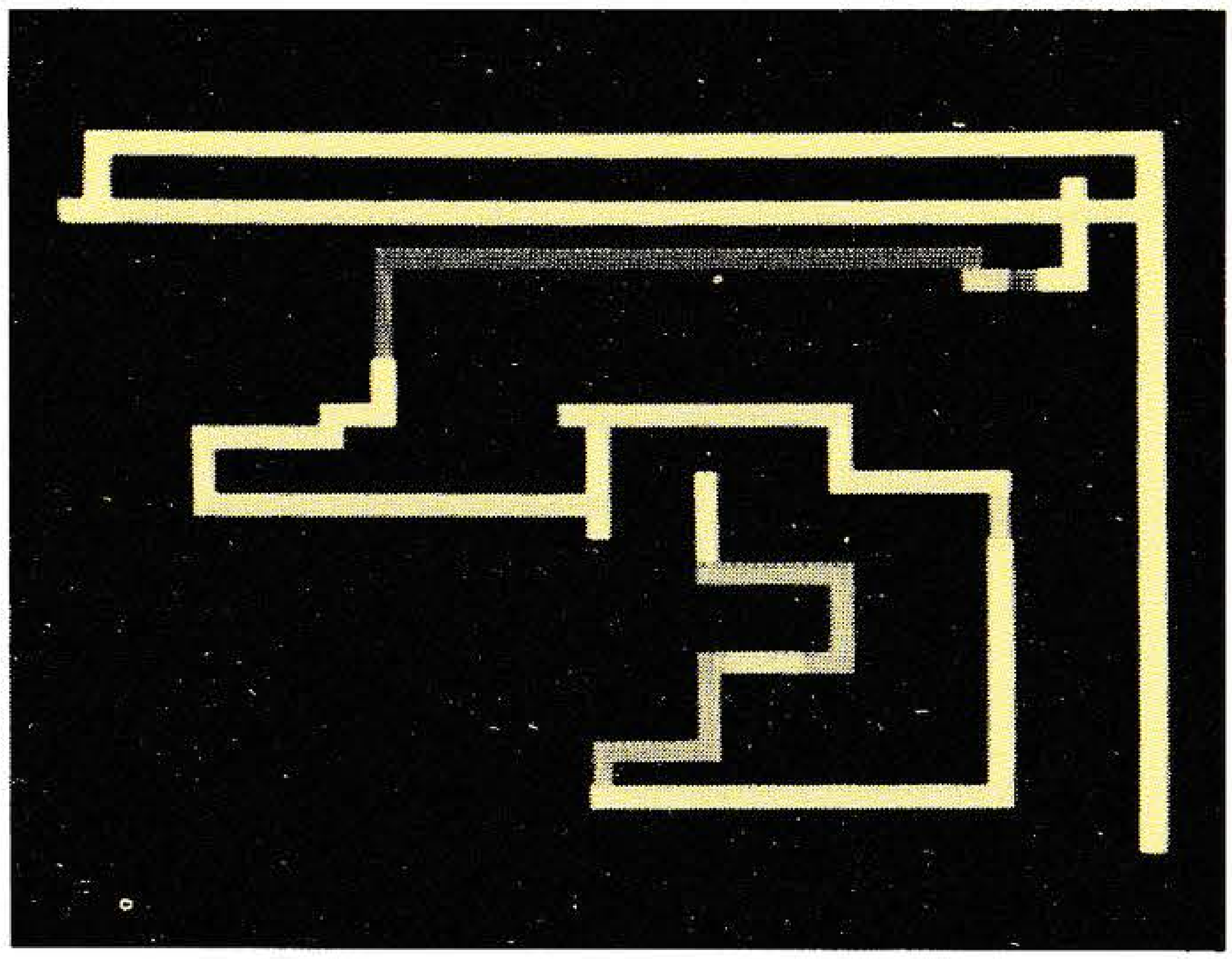


Figure 29

List 15

```
100 ' List   15
110 '
120 X=128 : Y=96 : C=2
130 '
140 PRINT "A,Z,X,C,ｽﾍﾟｰｽ ｷｰ ｦ ｵｼﾃﾈ"
150 FOR T=1 TO 500 : NEXT T
160 '
170 SCREEN 3 : COLOR 15,1,1 : CLS
180 PSET (X,Y),C
190   ON INSTR("cxza ",INPUT$(1)) GOSUB 220,230,240,250,260
200 GOTO 180
210 '
220 IF X+4>255 THEN RETURN ELSE X=X+4:RETURN
230 IF X-4<0   THEN RETURN ELSE X=X-4:RETURN
240 IF Y+4>191 THEN RETURN ELSE Y=Y+4:RETURN
250 IF Y-4<0   THEN RETURN ELSE Y=Y-4:RETURN
260 IF C=15 THEN C=2 ELSE C=C+1
270 RETURN
```

List 15a

```
140 PRINT "ｶｰｿﾙ ｷｰ, ｽﾍﾟｰｽ ｷｰ ｦ ｵｼﾃﾈ"
190   ON STICK(0) GOSUB 250,270,220,270,240,270,230,270
195   IF STRIG(0) THEN GOSUB 260
```

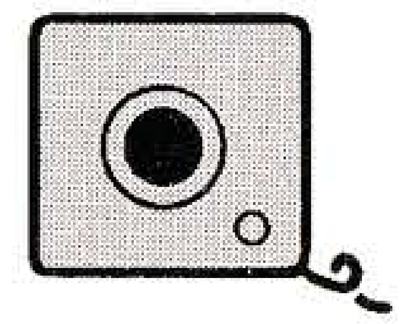
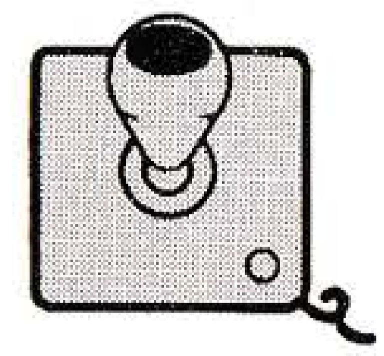
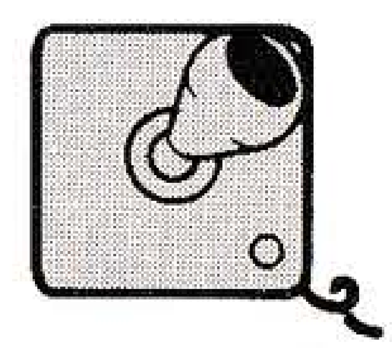
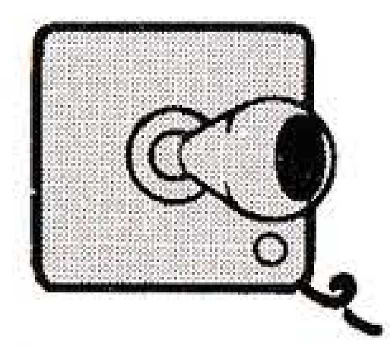
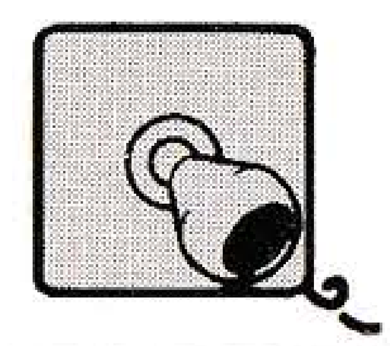
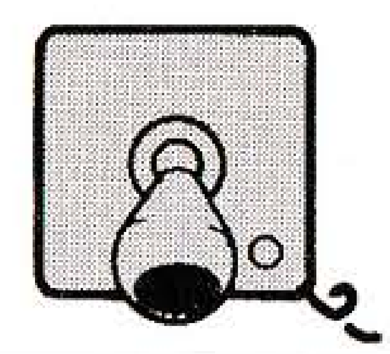
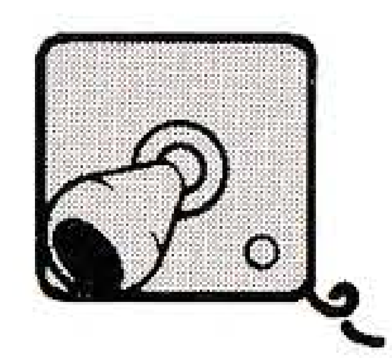
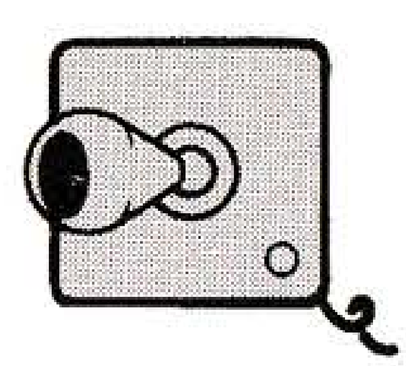
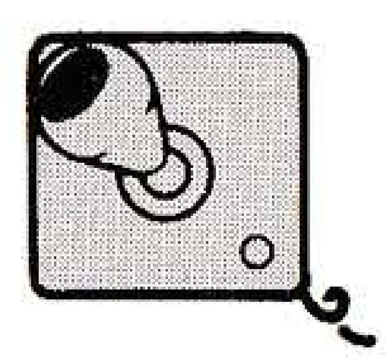
はマイナス4，すればよいのです．220行はXに4を足すサブルーチンです．ただし4を足すと画面右端からはみ出てしまう（つまりXが255を超える）場合には何もしないで戻ります．同様に，230，240，250行は画面からはみ出さない場合に，それぞれX−4，Y+4，Y−4を行うサブルーチンです．

260行は四角形の移動とは関係ありません．これは，色を決める変数Cを操作しています．それまでC=15だったなら，このサブルーチンの実行で次はC=2となり，そうでなければCは1だけ増えます．結局Cの値はこのサブルーチンを1回実行するごとに，2→3→4→5…13→14→15→2……と変化していきます．

そして，押されたキーによって5つのサブルーチンの中から1つを選び出して実行させている，プログラムの心臓とでもいうべき部分が190行です．今，どのキーが押されたのか，それはINPUT$（1）を調べるとわかります．INSTR関数を用いると，その文字が特定の文字列（ここでは"cxza　"）の何番目にあるかがわかります．さらに，その何番目の文字であるかという順番によって，ON～GOSUB命令が行き先を振り分けてくれるのです．たった1行の命令ですが，この190行は非常に中身の濃い仕事をしているんですよね．

190行のINSTR関数の中にある文字列を他のものに変えれば，別のキーの組み合わせで移動方向を指定することもできます．また，カーソルキー入力を判断するためには，特に専用の関数が用意されています．そこで*List 15*を*List 15a*のように

Table 1

	STICK(0)	STICK(1)
0	—	
1	↑	
2	↑ と →	
3	→	
4	→ と ↓	
5	↓	
6	↓ と ←	
7	←	
8	← と ↑	

変更してください.

190行を見てください，STICK(0)関数は，カーソルキーの押されている状態を0から8までの数で返してくる関数です. *Table 1* のように⇧が押されているときは1, ⇨が押されているときは3, ⇧と⇨の両方が押されているときは2, 何も押されていなければ0……を返してきます.

STICK 関数の他にも，MSX には STRIG 関数があります. この STRIG(0) 関数は，スペース バーが押されているときは−1，押されていないときは0を返します.

STICK (n)，STRIG (n) 関数の引数は共に0となっていますが，この n を1または2とするとジョイスティックによる判定ができます.

STICK (n) は，ジョイスティックをどの方向に倒しているのかを，STRIG (n) は，トリガボタンが押されているかどうかを調べる関数となります (*Table 1* 参照).

ジョイスティックを持っている人は，この引数を1にしてためしてください.

位置の判断

グラフィックが今，画面上のどこに位置しているのか，この判断は意外と重要な問題です. ある領域から外へ出してはいけない物体をどうすれば閉じこめておけるのか. アキレスが亀のシッポに触れたかどうか調べるにはどうするか. 要するにすべて座標の問題だといってしまえばそれまでですが，グラフィックは上下左右の幅を持っているため，簡単ではありません. 間違いなく，正確に，位置の判断をするためには，緻密な頭脳と注意深いプログラミングが必要なのです.

List 16 はその位置判断のささやかな例です. *List 16* を実行すると，まず乱数でゴールのラインが引かれ，そこに向かってランナーが走っていきます. ランナーの体の一部分でもゴールラインに接すれば，そこでレースは終わり，音楽が流れます (*Figure 30* 参照).

230〜300行は，ランナーの絵を描くサブルーチンです. ゴールラインを表す x座標は140行で決めて，G という変数に代入しておきます. ランナーが走ってきて，この G の位置を越えたらゴールイン. ランナーの位置は (X，Y) という座標で指定しますから，変数 X の値が G の値を超えたら終わりにすればいいかな，と考えるのはちと甘い.

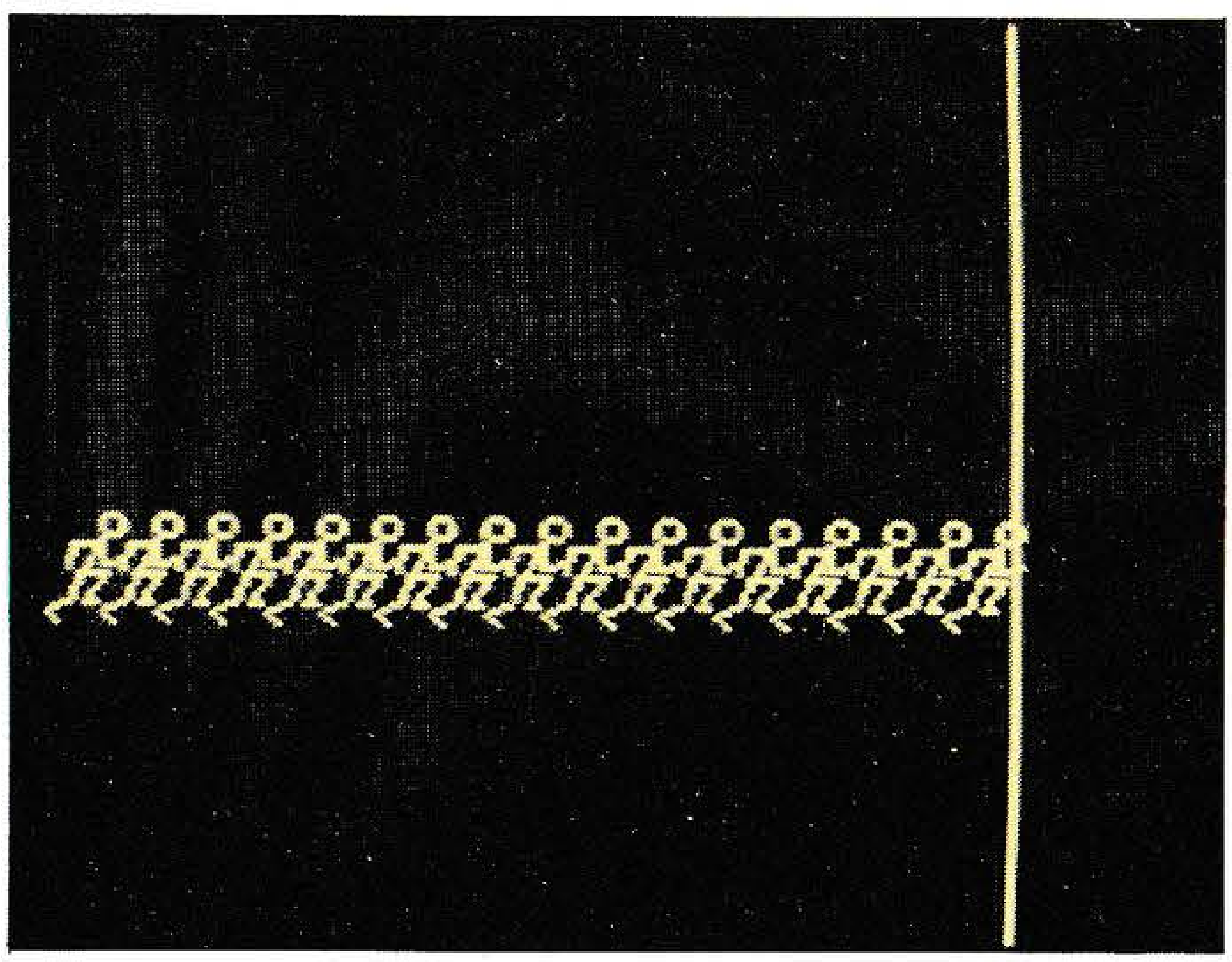


Figure 30

さきほどもいいましたがランナーのグラフィックは幅を持っています．その左端のx座標はX+2，右端のx座標はX+16です．このグラフィックが一部分でもゴールに接したときというのは，右端のX+16がゴールに達したときのことをいいます．すなわち，190行のようにX+16の値がGの値を超えたということをもって，ゴールインを判断しなければならないのです．

それでは応用問題をひとつ．「ランナーの全身がゴールラインを超えた時点で，ゴールインと判定する」という場合には，190行をどのように変えればいいでしょう？ はいそうですね． *List 16a* のように，IF THEN 命令の判断を変えればよいのです．

List 16

```
100 ' List   16
110 '
120 SCREEN 2 : COLOR 15,12,12 : CLS
130 R=RND(-TIME)
140 G=INT(RND(1)*100)+140
150 LINE (G,0)-(G,191),15
160 '
170 X=0 : Y=106 : C=15
180 GOSUB 230
190   IF X+16>G THEN 320
200   X=X+10
210 GOTO 180
220 '
230 CIRCLE(X+14,Y),2,C
240 LINE (X+13,Y+2)-(X+9,Y+10),C
250 DRAW "S4A0 BM=X;,=Y; BM+6,+9 C=C;"
260 DRAW "M+1,-5 M+5,+0 M+0,+4"
270 DRAW "M+4,-1 BM-8,+3 M-1,+5"
280 DRAW "M-5,+3 M+2,+2 BM+4,-10"
290 DRAW "M+5,+0 M-4,+6 M+2,+0"
300 RETURN
310 '
320 PLAY "T128O6L8GGEFGGCCL16DEFGL8FED"
330 FOR T=1 TO 1500 : NEXT T
340 END
```

List 16a

```
190   IF X+2>G THEN 320
```

スプライト機能を使おう

LINE 命令や DRAW 命令を使って図形を描きその図形を移動させる場合には，常に描いては消すか，消しては描くかしなければなりませんでした．この処理が面倒だと思わない人は,1人でプログラムが書ける人か,よほどガマン強い人でしょう,MSX には，動くグラフィックに最適なスプライト機能というものが存在するのですから，使わない手はありません．しかし今まで学んできた「動きのあるグラフィック」のテクニックは，スプライト機能があるから必要ないというのではありません．共に一長一短があるもので，それを必要に応じて使いわけるべきだと筆者は考えます．

スプライト画面が，32枚のガラス板のような画面で，このガラス板1枚につき1つのスプライトパターンを置くことができ，このパターンを置き直すだけで図形の移動が可能であることは,1章で説明したとおりです．

実際にスプライト機能を使うには，次のステップをふむ必要があります（*Figure 31* 参照）．

①スプライトパターンを考える．

スプライトパターンは，8×8ドットまたは16×16ドットのうちの1つしか選択できないので，このパターンの大きさ，細かさによって8×8ドットか16×16ドットでパターンを考える必要があります．

②スプライト機能で表示する図形のデータを文字列として定義する．

これは，MSX を買ったときに付いてくる MSX-BASIC リファレンスマニュアルで SPRITE$ を引いてみてください．スプライトパターンの定義例としてスプライトパターンのデータを作る方法が詳細に書かれています．

……したがって，パス．

③ SCREEN 命令でスプライトパターンを表示する画面とその大きさを指定する．

スプライトパターンを利用できる画面モードは，SCREEN 1～3 に限られます．どの画面モードを利用するのかよく考えて選択してください．さらに *Table 2* のようにスプライトパターンは，表示するときに2倍に拡大することもできます．これは，SCREEN

Table 2

SCREEN , 〈スプライトサイズ〉	
SCREEN ,0	8×8画素標準 （8×8ドットの大きさ）
SCREEN ,1	8×8画素拡大 （16×16ドットの大きさ）
SCREEN ,2	16×16画素標準 （16×16ドットの大きさ）
SCREEN ,3	16×16画素拡大 （32×32ドットの大きさ）

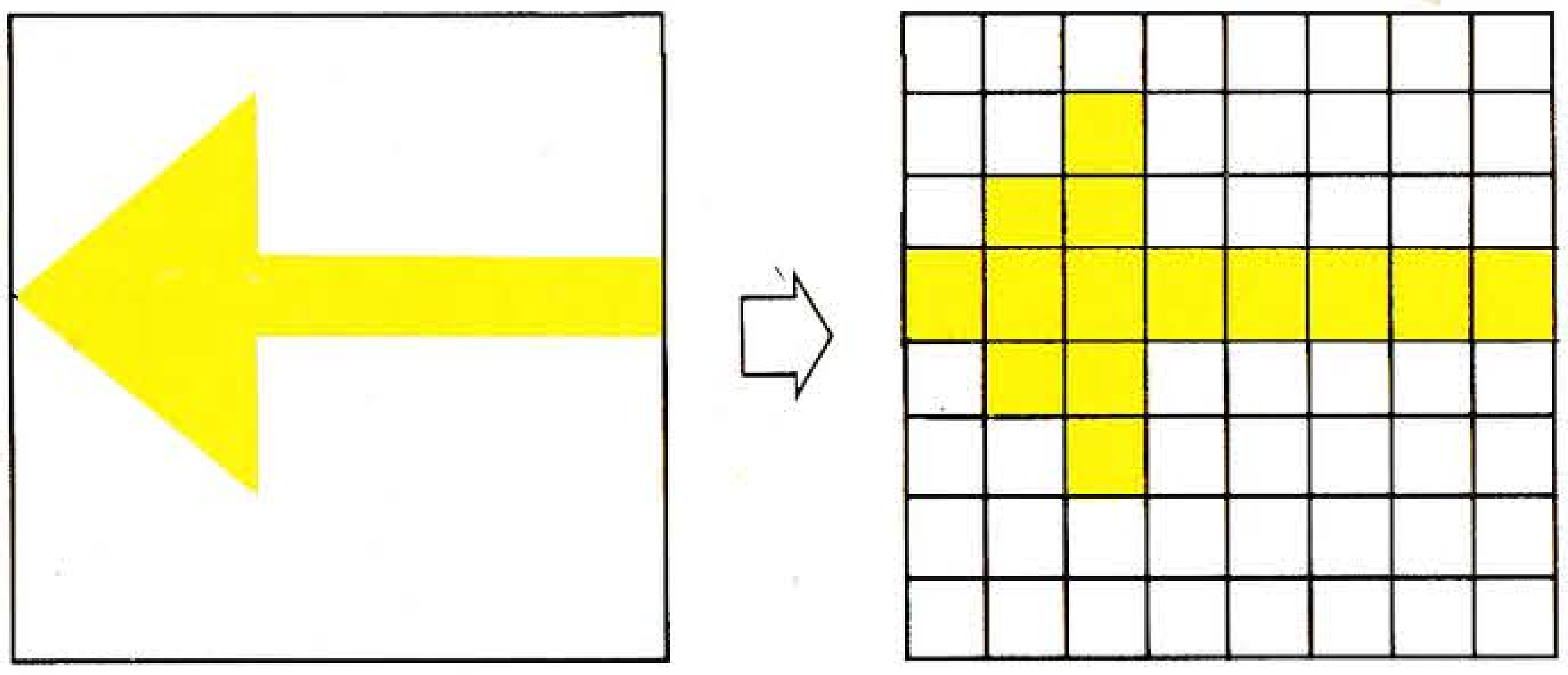
命令の第2パラメータで設定します．

④スプライトパターン番号に，スプライトパターンデータを割り当てる．

スプライトパターン・データは，すでに文字列として定義されています．この文字列は，スプライトパターン番号で表すことにします．このスプライトパターン番号は，スプライト面にパターンを表示する際利用します．

⑤スプライト面にスプライトパターンを表示する．

スプライト面（0〜31）に，スプライトパターン番号で表されるスプライトパターンを置きます．



2進

& B 0 0 0 0 0 0 0 0
& B 0 0 1 0 0 0 0 0
& B 0 1 1 0 0 0 0 0
& B 1 1 1 1 1 1 1 1
& B 0 1 1 0 0 0 0 0
& B 0 0 1 0 0 0 0 0
& B 0 0 0 0 0 0 0 0
& B 0 0 0 0 0 0 0 0

① スプライトパターンを考える

② スプライト機能で表示する図形データを文字列として定義する

10進	文字列
0	CHR$(0)
32	CHR$(32)
96	CHR$(96)
255	CHR$(255)
96	CHR$(96)
32	CHR$(32)
0	CHR$(0)
0	CHR$(0)

文字列に定義する

A$=CHR$(0)+CHR$(32)+CHR$(96)+
CHR$(255)+CHR$(96)+CHR$(32)+
CHR$(0)+CHR$(0)

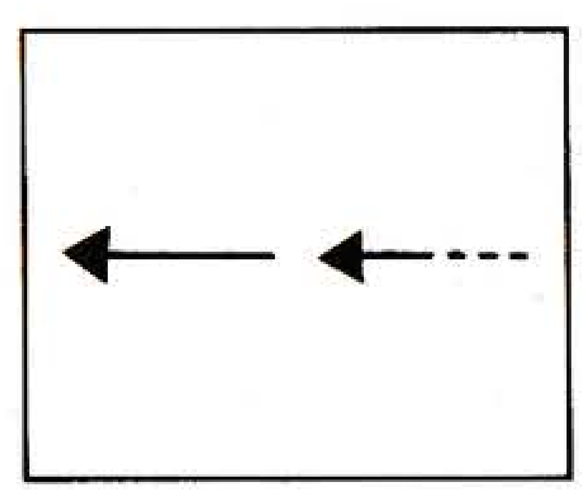
SCREEN 2,1 RETURN

スプライトパターンを表す文字列

SPRITE$(1)=A$

スプライトパターン番号

画面



③ SCREEN命令でスプライトパターンを表示する画面とその大きさを指定する

④ スプライトパターン番号に，スプライトパターンデータを割り当てる

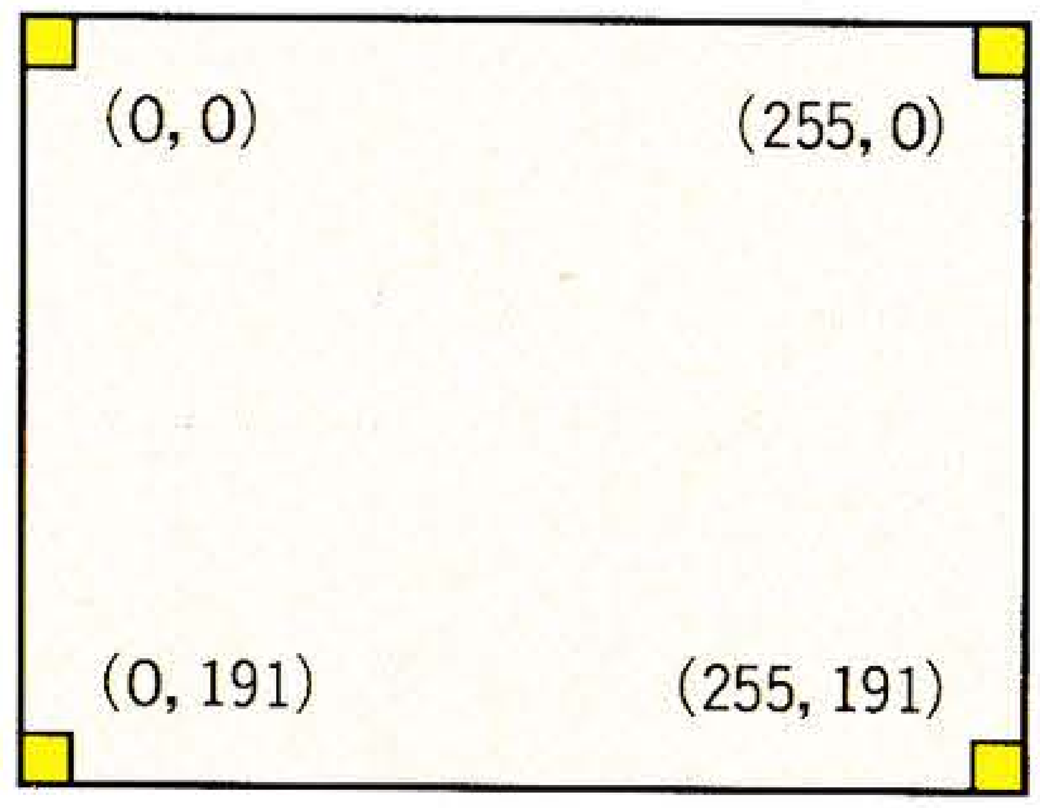
⑤ スプライト面にスプライトパターンを表示する

Figure 31

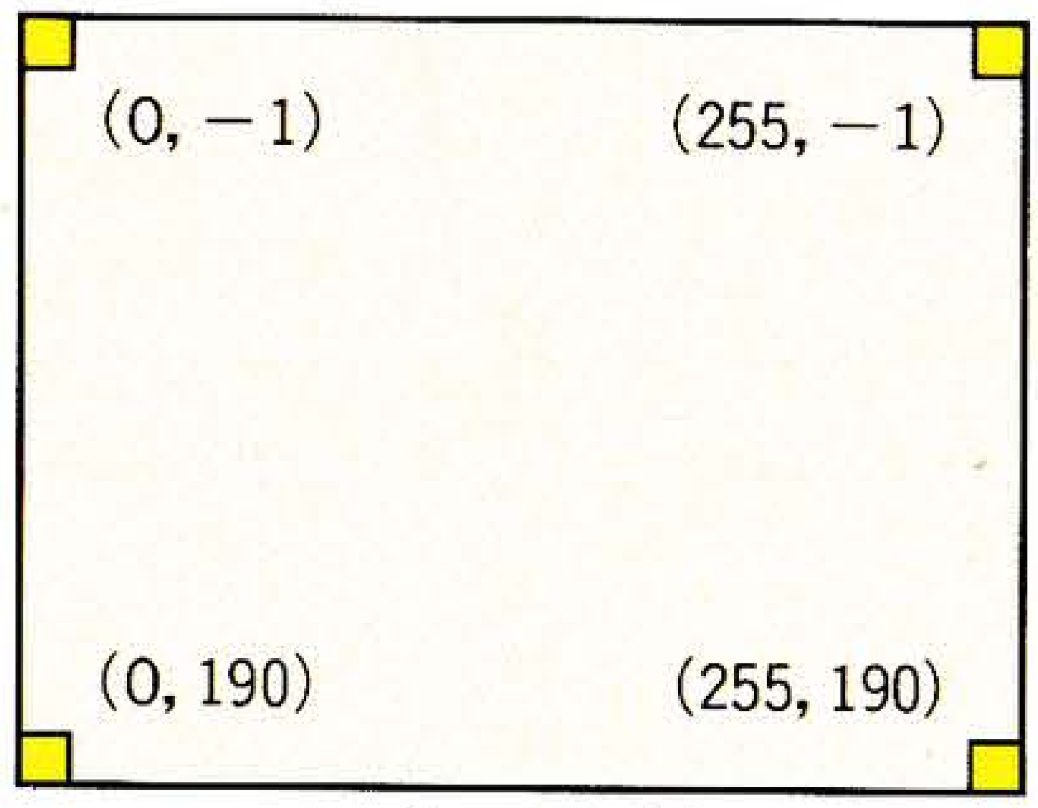
この手順で，実際にプログラミングしたものが *List 17* です．もうお分りですね．220～290の DATA を書き換えてみてください．

PUT SPRITE 命令を実行する際に，1つだけ注意してほしいことがあります．スプライト画面の座標は，*Figure 32* のように左上すみが (0，0) ではなく (0，−1) であることです．たとえば，グラフィック座標に LINE，CIRCLE 命令で描かれた図形と，スプライト機能によって描いた図形を重ねたいときなど，この1ドット分のずれを考慮しなくてはなりません．

さて，この章では，そもそもグラフィックを動かすことが目的でした．そのための準備とも言えるスプライト機能についての



グラフィック座標



スプライト座標

Figure 32

List 17

```
100 ' List   17
110 '
120 RESTORE 220
130 FOR I=1 TO 8
140   READ D$ : A$=A$+CHR$(VAL("&B"+D$))
150 NEXT I
160 '
170 SCREEN 2,1 : COLOR 15,4,4 : CLS
180 SPRITE$(1)=A$
190 PUT SPRITE 0,(100,90),15,1
200 I$=INPUT$(1):END
210 '
220 DATA 00000000
230 DATA 00100000
240 DATA 01100000
250 DATA 11111111
260 DATA 01100000
270 DATA 00100000
280 DATA 00000000
290 DATA 00000000
```

説明です．理解できましたね．では，いよいよスプライト機能を使った「動くグラフィック」を行ってみることにします．

List 18 を実行してみてください．何と矢印が速く動くことでしょうか．これがスプライト機能の実力です．

List 18 の120行を見てください．スプライトパターン・データは2進数で表記せずに *Figure 31* のように，横一列を10進数に変換し，CHR$ (n) によって文字列として扱っています．180〜230行は，バックに散りばめるブロックを描いている部分です．そして，スプライトパターンの移動は，290〜320行の4行で行っています．

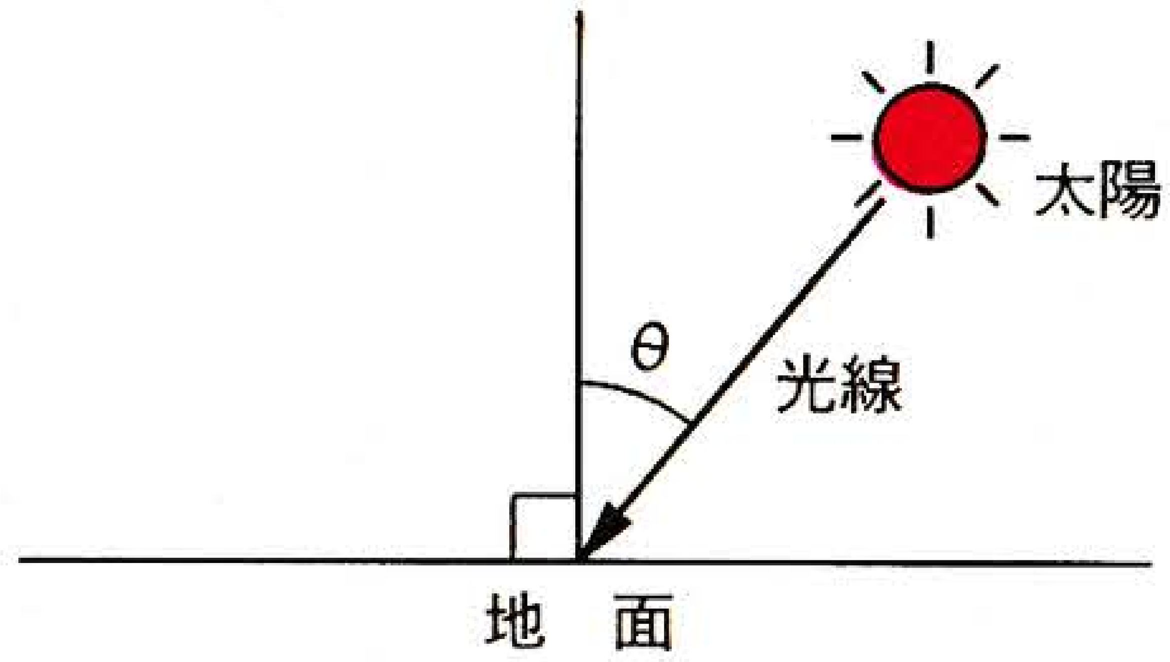
以上がスプライト機能を利用した，動くグラフィックの説明でした．

スプライト機能に関しては，ON SPRITE GOSUB という割り込み命令も用意されています．これは，4章に登場します．よろしく．

List 18

```
100 ' List   18
110 '
120 A$=CHR$(0)+CHR$(32)+CHR$(96)+CHR$(255)+CHR$(96)+CHR$(32)+CH
R$(0)+CHR$(0)
130 X=240 : Y=90 : SC=11
140 '
150 SCREEN 3,1 : COLOR 15,1,1 : CLS
160 SPRITE$(0)=A$
170 '
180 FOR I=1 TO 50
190   PX=INT(RND(1)*64)*4
200   PY=INT(RND(1)*48)*4
210   C=INT(RND(1)*14)+2
220   PSET (PX,PY),C
230 NEXT I
240 '
250 PUT SPRITE 1,(X,Y),SC,0
260 PLAY "T128L2O3A"
270 FOR T=1 TO 1000 : NEXT T
280 '
290 PUT SPRITE 1,(X,Y),SC,0
300   X=X-8
310   IF X<10 THEN I$=INPUT$(1):END
320 GOTO 290
```

暑さ寒さはコサインで

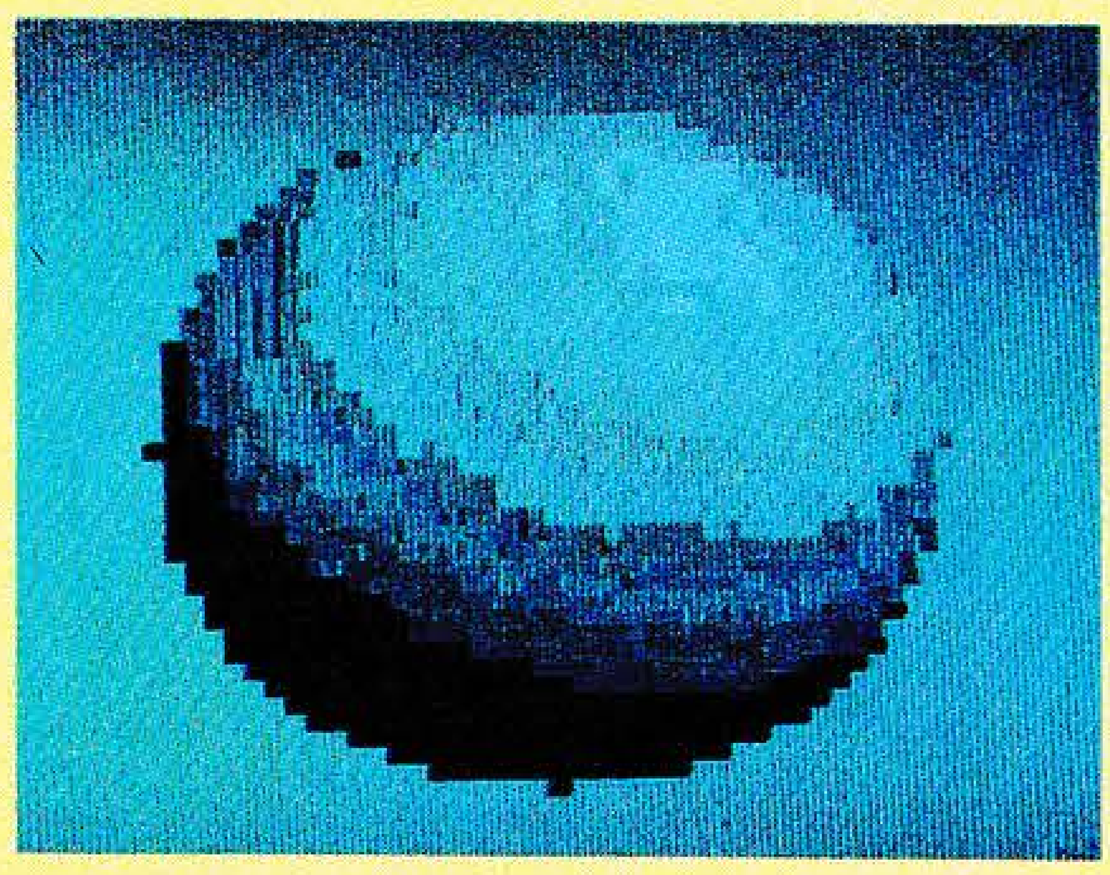


気候というものは土地によって様々です．灼熱の太陽照りつける熱帯の国々がある一方で，南極や北極という氷に閉ざされた厳寒の地もまた，現に存在しているのです．同じ地球の上にあって，なぜこのような違いが生ずるのでしょうか．

このような気候の差は，もっぱら太陽光線の入射角度に起因しています．太陽が頭の真上から照らせば気温は高くなり，太陽の位置が低ければ気温も低くなります．なぜなら光線の入射方向と垂直との角度をθとすれば，地面が受ける入射エネルギーの量はcosθに比例するからです（図参照）．

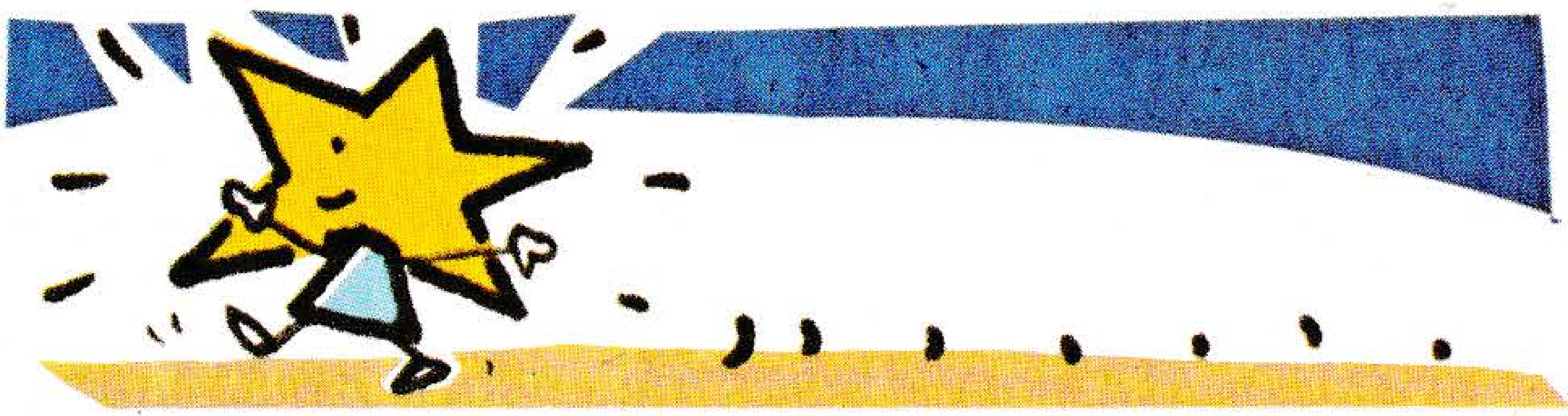
さて，いったん数学的な関係式がわかってしまえば，それを目に見える形で表示するのはコンピュータ・グラフィックスの最も得意とするところ．まずカラーバランスつまみを回して，画面を白黒にしてからこのプログラムを実行してください．右上方向からの光線に照らされた球体が次第に姿を現します．

```
100 ' List Ex.2
110 '
120 R=80 : UX=1 : UY=2 : UZ=2
130 '
140 U=SQR(UX*UX+UY*UY+UZ*UZ) : UX=UX/U : UY=UY/U : UZ=UZ/U
150 DIM C(15)
160 FOR I=0 TO 14 : READ C(I) : NEXT I
170 '
180 SCREEN 3 : COLOR 1,3,3 : CLS
190 '
200 FOR Y=-R TO R STEP 4
210   FOR X=-R TO R STEP 4
220     Z2=R*R-X*X-Y*Y
230     IF Z2<0 THEN 280
240       Z=SQR(Z2)
250       C=(X*UX+Y*UY+Z*UZ)/R
260       E=INT(C*14+RND(1)) : IF E<0 THEN E=0
270       PSET (128+X,96-Y),C(E)
280   NEXT X
290 NEXT Y
300 I$=INPUT$(1):END
310 '
320 DATA 1,4,12,6,13,2,5,8,3,7,9,10,14,11,15
```



【3章】

アートへのアプローチ



直線で模様を描こう
MUG CUPS
スペースシャトル
リンゴを描く
落葉と星型
Stardust Night
円は異なもの味なもの
自分だけの曲線を描こう
ヘリコイド曲線を描く
ネコでも回そうか?
3次元グラフィックス?
文字の変形

本章ではコンピュータによるアートに挑戦します.

近年，我等が MSX をはじめとする高性能パソコンが出現し，これまでは高嶺の花であったコンピュータ・グラフィックスが私たちにも手軽に楽しめるようになりました. なかでも，本章に掲載したような世人の目を楽しませる壮麗な，あるいは小意気に洒落たグラフィック作品が，わずかな手間で描けるこの楽しさ……. 美しい作品を描くためのテクニックのその入口あたりを種々取りまぜて見てもらうことにいたします. 月並ないい方ですが，これを基礎として皆さん自身，いろいろなグラフィックに取り組んでいただきたい. 筆者らはそう希望しております.



直線で模様を描こう

本章のテーマであるアートという言葉に何やら高尚な，そしてむずかしそうな印象を感じた方，いえいえそんなに身構える必要はありません．

何はともあれ，まず直線を一本引くことから始めましょう．それだけでも美しい模様は描けるのです．

ここに挙げた 2 つのプログラムを見てください．扱っているグラフィックは，どちらも LINE 命令だけで描かれていますがちょっとすてきな模様となっていると思いませんか？

List 1a では，横 XB から XE，縦 YBからYEの四角形の各辺上を直線の始点と終点を移動させます．190行で引いている直線を例にとれば，始点は四角形の底辺を S 分割して，左から右に XS ドットずつ動いていき，終点は四角形の右辺を下から上に向けて，YS ドットずつ動いていきます（下図参照）．この始点と終点を順に結ぶと，右下の角に相当する部分が描きあがります．同様に，200～220行までの LINE

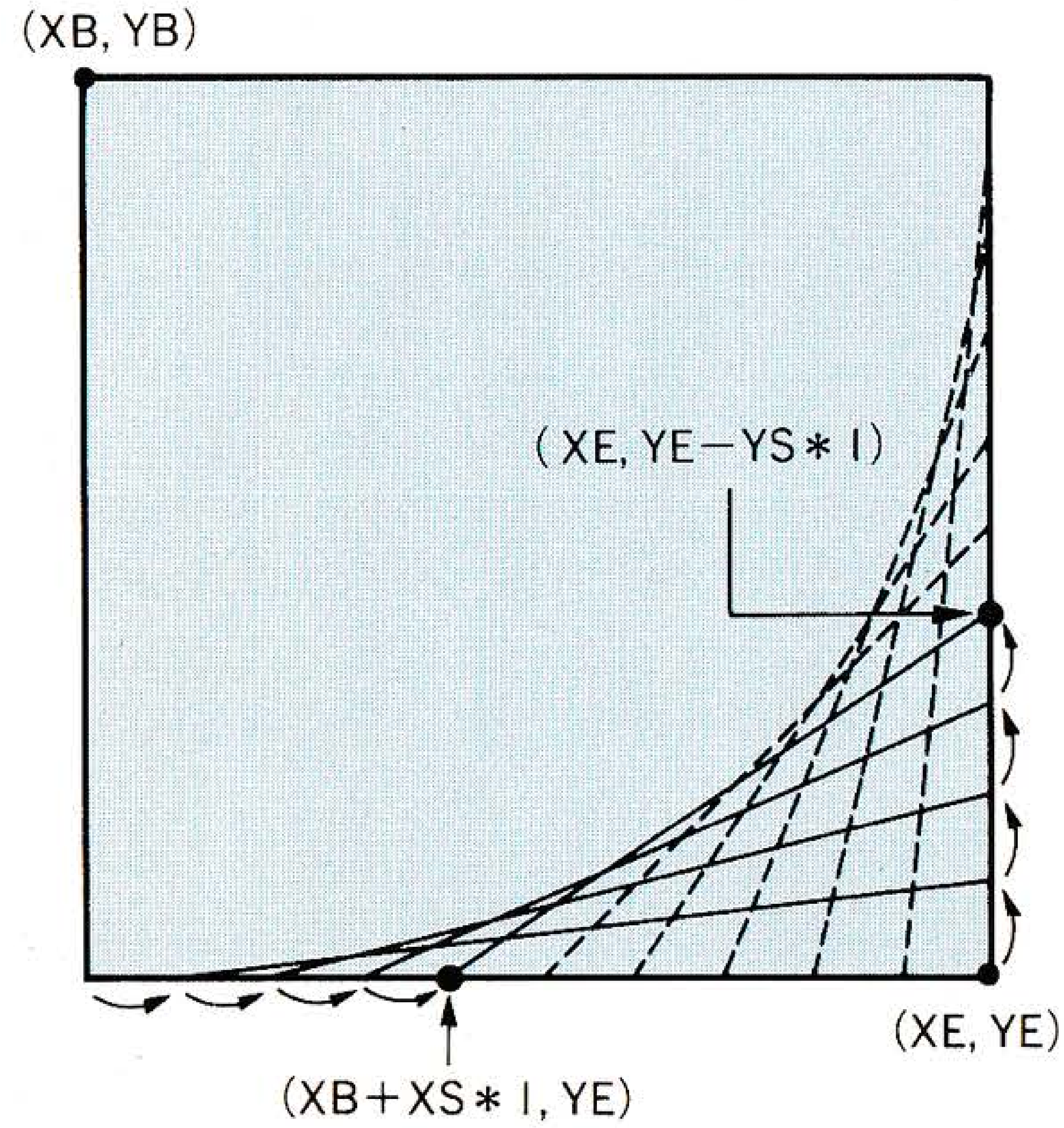
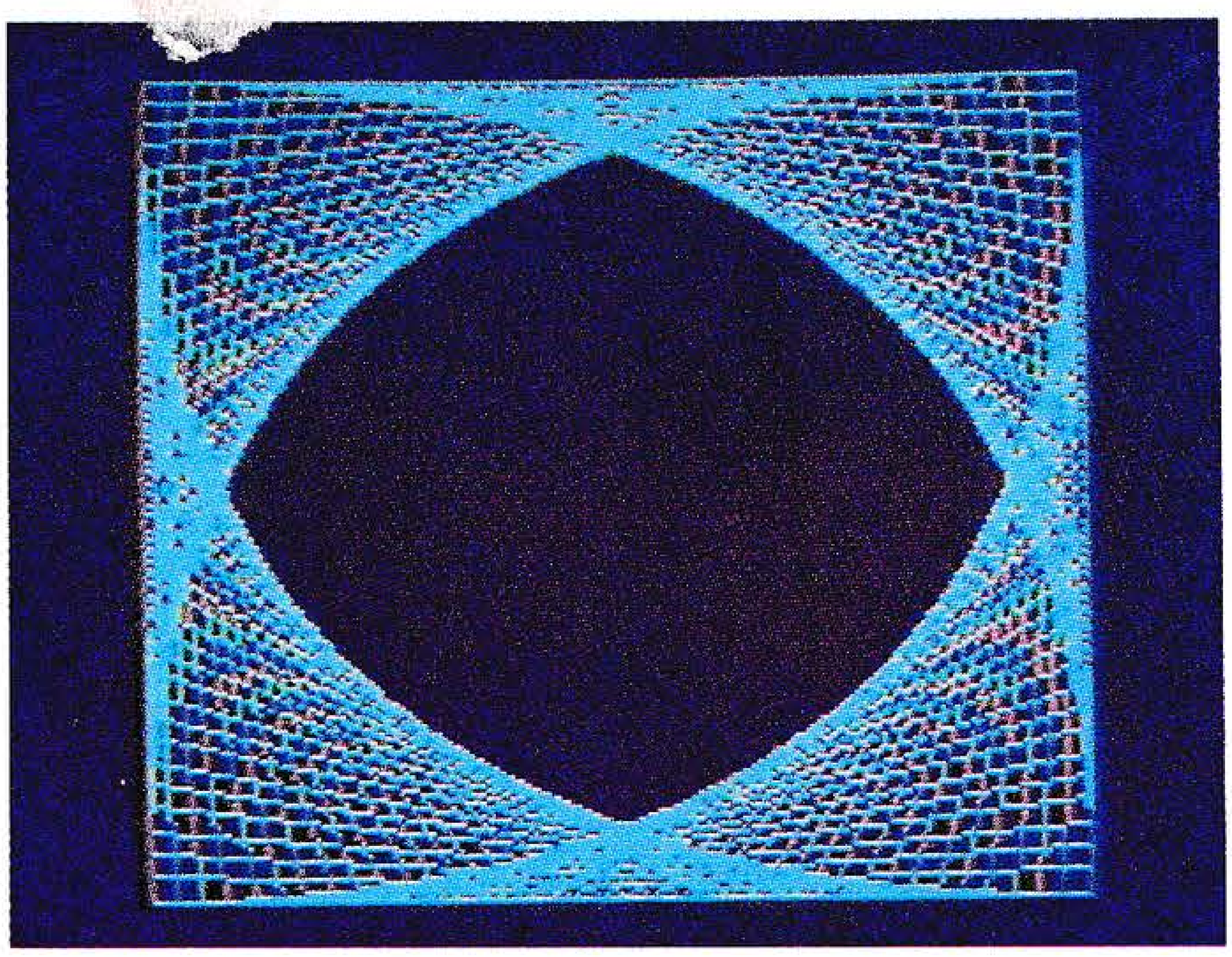
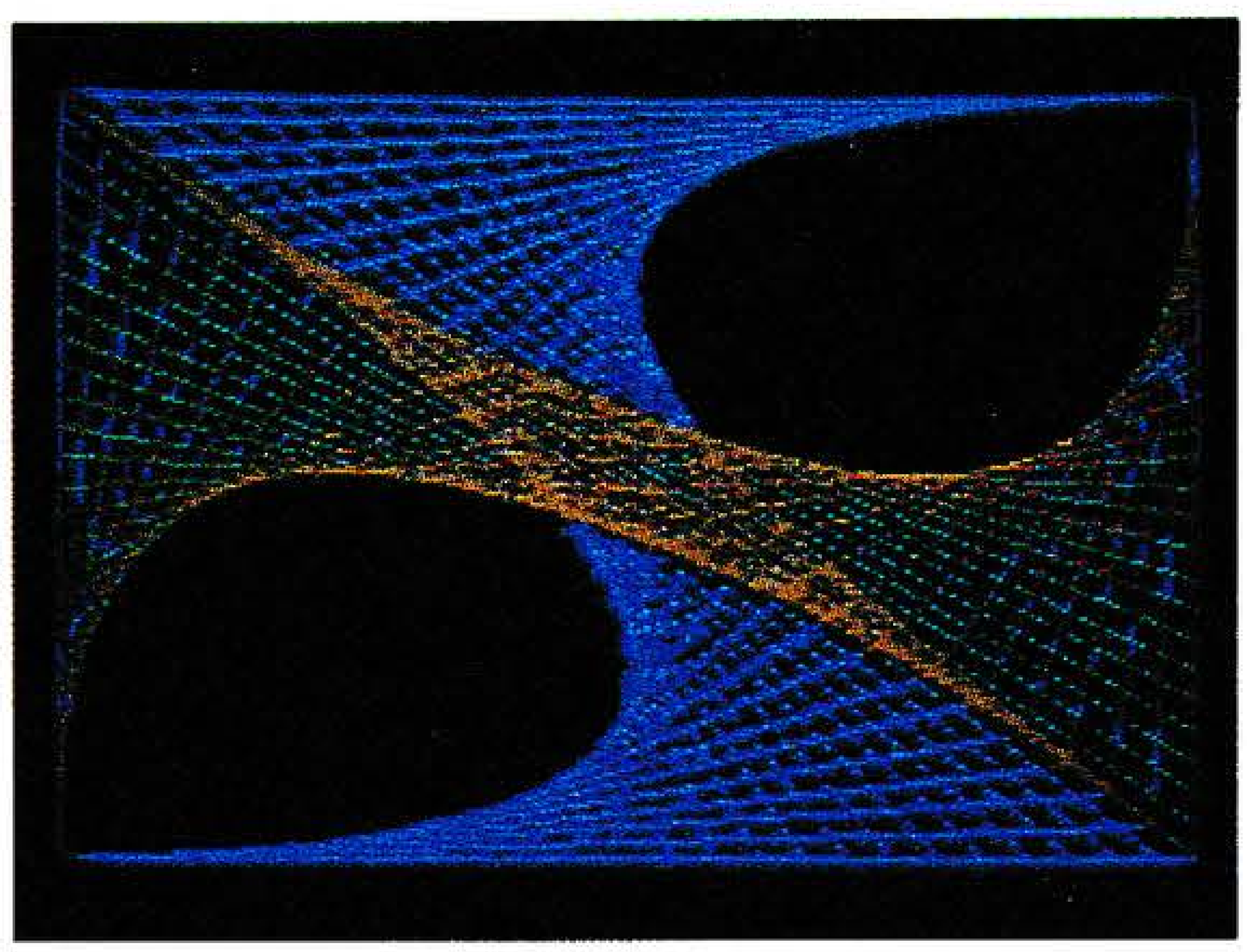


Figure 1





命令が，四角形の右上，左上，左下の角を描いていくわけです．

List 1b でも，やはり直線の始点と終点を動かしていますが，今度は四角形の辺と対角線を結んで線を引いています．

どちらのプログラムも変数 S や XE，YE などの値を変えると画面が変わります．変化する模様を楽しんでください．

List 1a

```
100 ' List   1a
110 '
120 XB=32   : YB=0
130 XE=223 : YE=191
140 S=36    : C=15
150 XS=(XE-XB)/S : YS=(YE-YB)/S
160 '
170 SCREEN 2 : COLOR 15,13,13 : CLS
180 FOR I=0 TO S
190   LINE (XB+XS*I,YE)-(XE,YE-YS*I),C
200   LINE (XE-XS*I,YB)-(XE,YE-YS*I),C
210   LINE (XE-XS*I,YB)-(XB,YB+YS*I),C
220   LINE (XB+XS*I,YE)-(XB,YB+YS*I),C
230 NEXT I
240 DM$=INPUT$(1):END
```

（120〜130）グラフィックを表示する範囲

（140）……各辺を分割する数とグラフィックの色

List 1b

```
100 ' List   1b
110 '
120 XB=0    : YB=0
130 XE=255 : YE=191
140 S=25 : C1=4 : C2=6
150 XS=(XE-XS)/S : YS=(YE-YB)/S
160 '
170 SCREEN 2 : COLOR 7,1,1 : CLS
180 FOR I=0 TO S
190   LINE (XE-XS*I,YB)-(XB+XS*I,YB+YS*I),C1
200   LINE (XB+XS*I,YE)-(XE-XS*I,YE-YS*I),C1
210 NEXT I
220 FOR I=0 TO S
230   LINE (XE,YB+YS*I)-(XE-XS*I,YE-YS*I),C2
240   LINE (XB,YE-YS*I)-(XB+XS*I,YB+YS*I),C2
250 NEXT I
260 DM$=INPUT$(1):END
```

MUG CUPS

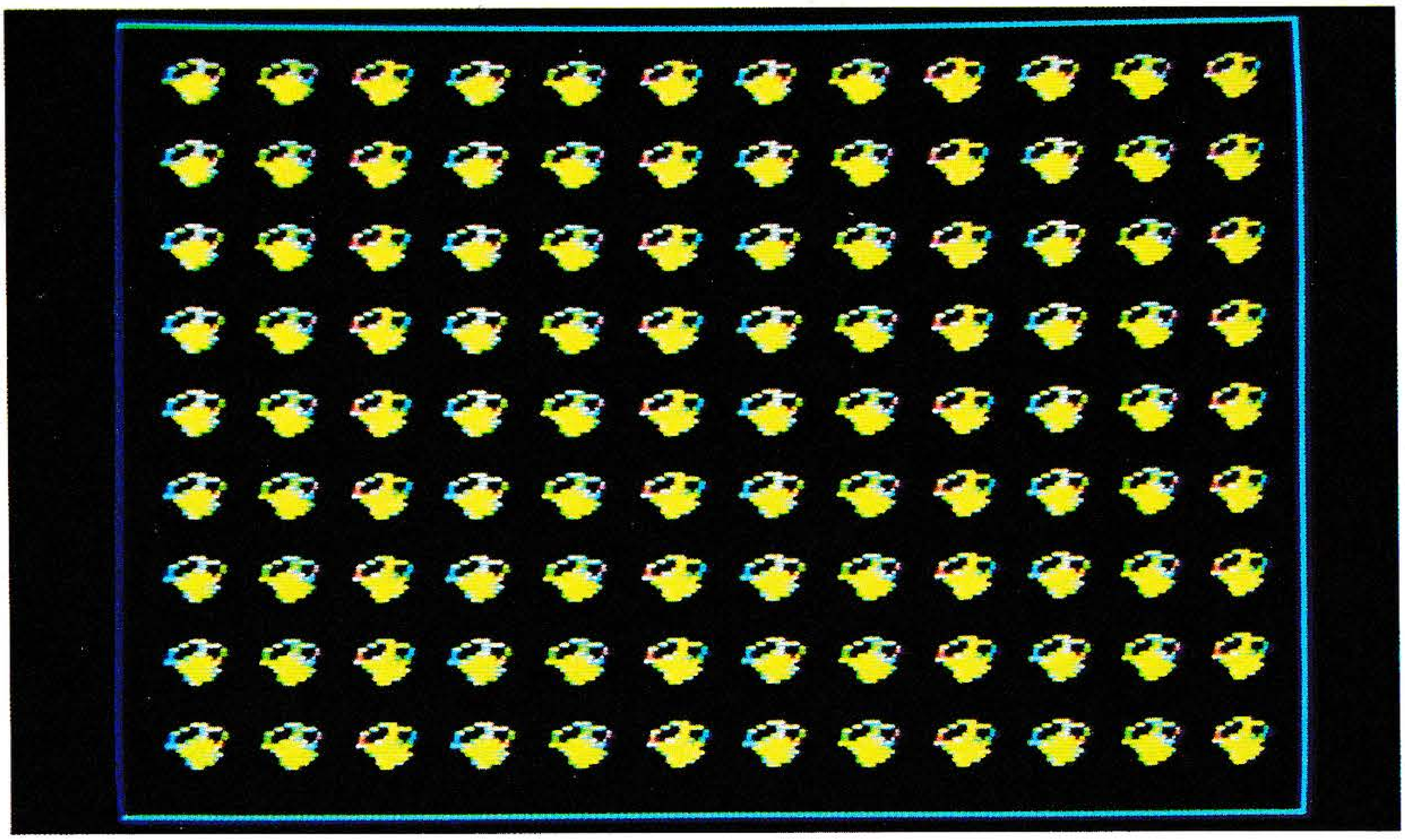
ひとつの図形を繰り返し描くことによって作り出される模様は，布地や壁紙などに数多く見受けられます．ここでは，かわいいマグカップを繰り返し描き，模様を作ってみましょう．

このプログラムは，190〜230行の2重のFOR〜NEXTループが，マグカップを描く始点の位置を決めています．そして，260〜330行のサブルーチンが，その始点をもとに，ひとつのマグカップを描きます．

350行以降のデータは，(0，0) を原点と考えたときの図形の各点の座標を示しています．なお，最後のデータ（−1，−1）は座標ではなく，データの終わりを示すために任意に決めた数で，この数はデータの中に出てきそうもない数であれば，何を使ってもかまいません．

270行では，マグカップを描き出すためのはじめの点をPSET命令によって打ち，300行のLINE命令が次々と連続して直線を引いていって，マグカップを完成させます．ちゃんと原点(X，Y)に，読み取ったデータA，Bを足して出力していますね．

AとBがともに−1になったら，そこでデータが最後だと判断し，線を延ばすのをやめます．このような形でプログラムを組



んでおくと，データを変更してもその個数を数えなおす必要がありません．

240行では，何かキーが押されるまで次のプログラムが実行されないようにして，プログラム終了時に画面が SCREEN 1 のモードに戻ってしまうのを防いでいます．

このプログラムに使われているマグカップのデータは，Appendix に掲載してあるデザインシートを使って，鉛筆で描いた下絵から読み取りました．読みとった座標データは，350～370行のように DATA として蓄えておく必要があります．

このように，コンピュータ・グラフィックスで望みの形を作り出す場合には，まずデザインシートに描いてみるという，基本的作業をめんどうくさがってはいけません．遠回りのように思えても，これが一番確実です．

List 2

```
100 ' List   2
110 '
120 XB=10  : YB=5
130 XE=230 : YE=170
140 XS=20  : YS=20 ……x軸方向に20，y軸方向に20ドットずつ離してカップを描くように設定する
150 C=11
160
メインルーチン
170 SCREEN 2 : COLOR 15,1,1 : CLS
180 LINE (0,0)-(255,191),7,B
190 FOR Y=YB TO YE STEP YS
200   FOR X=XB TO XE STEP XS
210     GOSUB 260
220   NEXT X
230 NEXT Y
240 DM$=INPUT$(1):END
250 '
サブルーチン
260 RESTORE 350
270 READ A,B : PSET (X+A.Y+B).11 ……始点を打つ
280 READ A,B
290   IF A=-1 AND B=-1 THEN 320……データが終わりかどうかの判定
300   LINE -(X+A,Y+B),11 ……線を結び，延ばす
310 GOTO 280
320 PAINT (X+10,Y+10),11
330 RETURN
340 '
350 DATA 2,9, 5,9, 9,6, 10,4, 8,4, 3,6, 3,9
360 DATA 4,10, 7,14, 11,13, 13,11, 10,7, 11,5, 13,5
370 DATA 14,7, 13,9, 11,9. 10,7, 10,4, -1,-1
```

スペースシャトル

“MUG CUPS”のプログラムでは，一筆描きでできる図形のデータをDATAとして蓄えておく方法を示しました．それでは，何本もの線で構成されている図形を描くにはどうしたらよいのでしょうか．ここに一例を示しました．

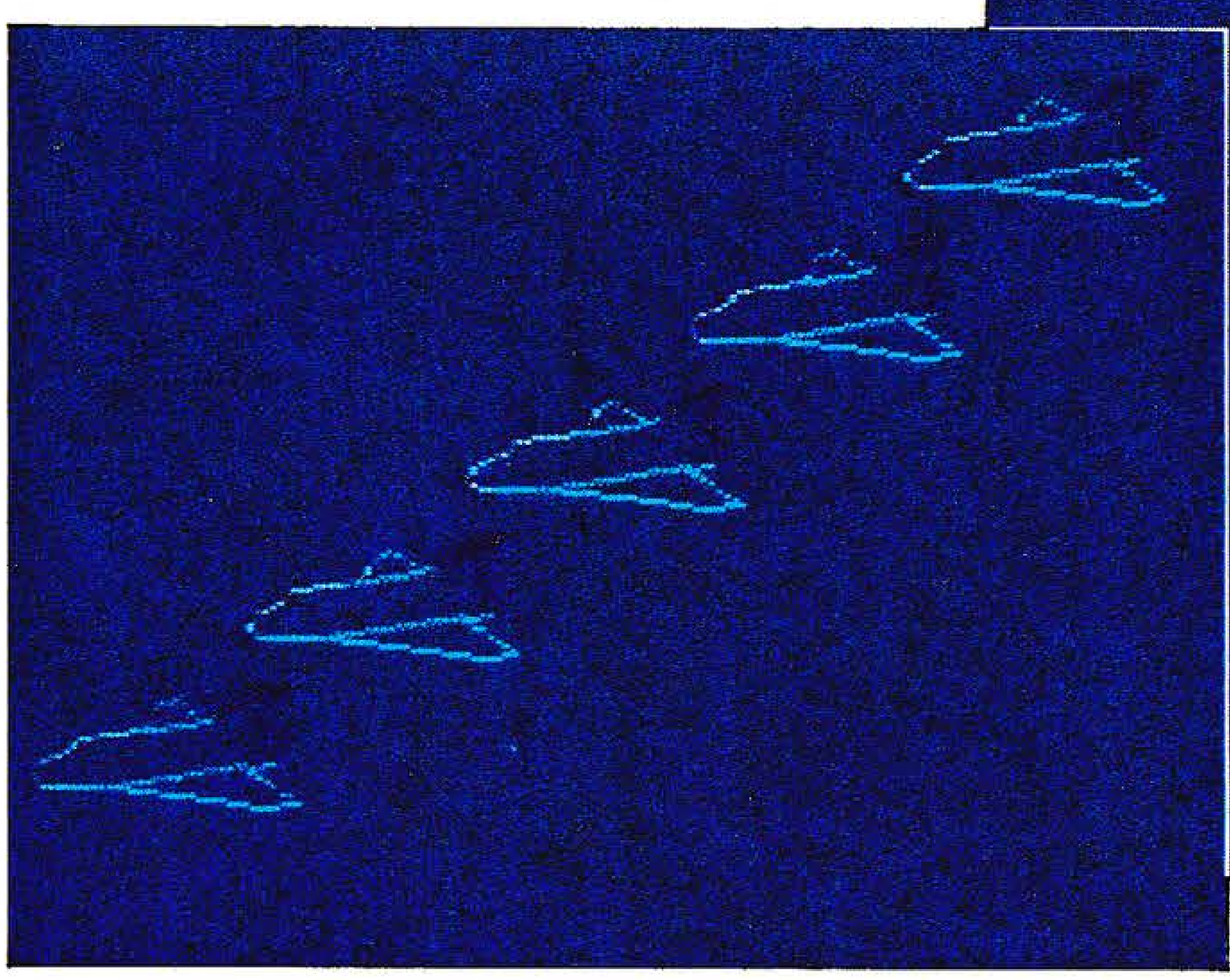
180〜210行はループになっています．190行の計算で図形の原点を決めています．250〜340行が，スペースシャトルを描くサブルーチンです．

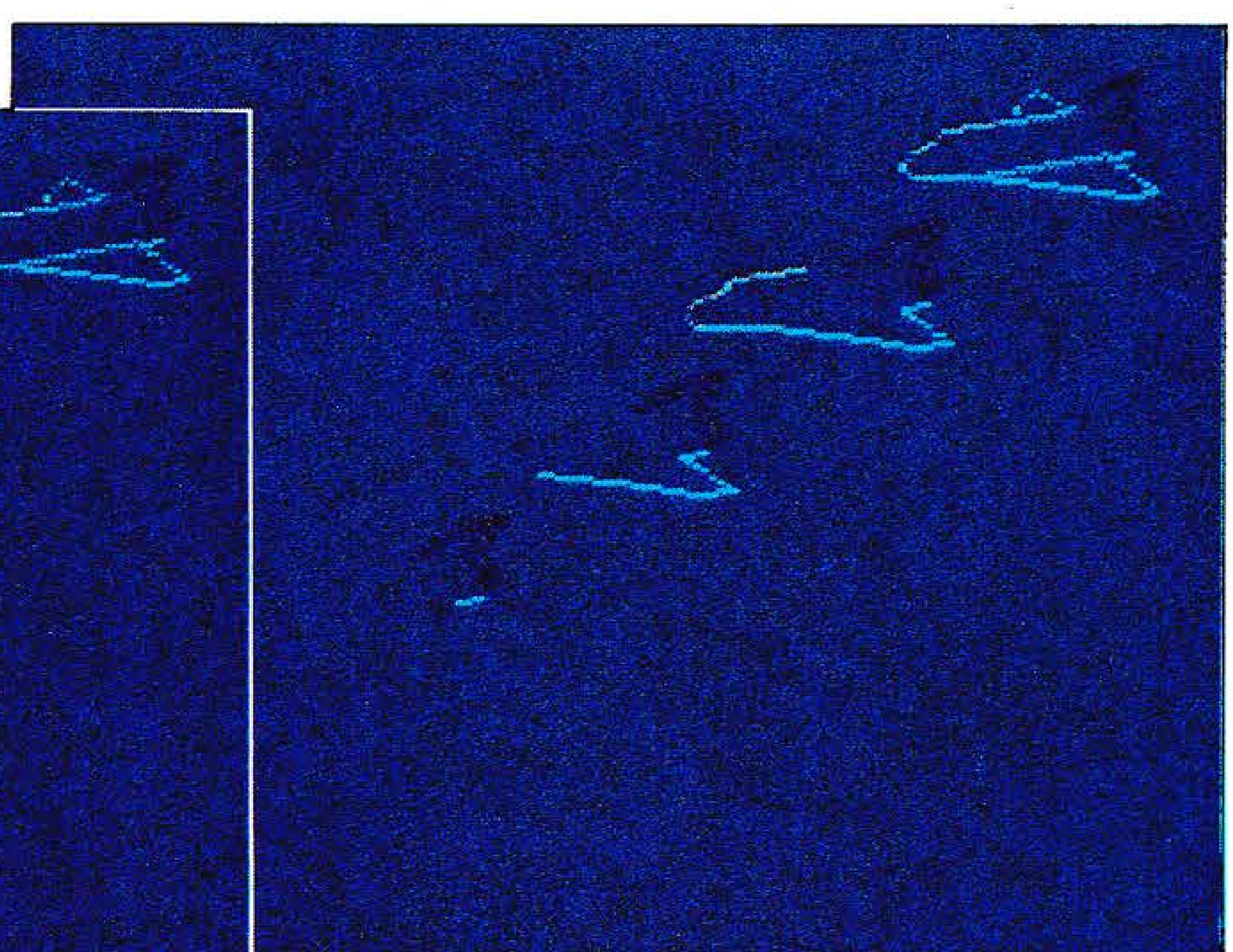
サブルーチン内で使われているデータは，やはり，座標（0，0）を原点とする相対的なものです．しかし，その数値は文字変数として読み取っています．なぜなら，データの区切りに“@”や“／”のマークを使うからです．ここでは“／”のマークを線の切れ目として，“@”のマークをデータの終わりとして使っています．

その他のデータは3つ1組で，順に，x 軸方向の座標，y 軸方向の座標，直線の色，となっています．データは読み取ったあとに，VAL 関数で文字データから数値データに変換して利用します．

280行や，それ以降で見られる FG という変数は，1つ前に引いた直線が，次に引く直線とつながるものかどうかを示す目印です．FG の値は，A$ が“／”を読み込んだときだけ−1となり，他は 0 となります．

もし FG＝−1ならば，その1つ前の直線と，今読み込んだ座標（X，Y）は，まったくつながりを持たず，次の直線の始点となるわけです．320行の





PSET(X,Y),C

という命令で，その始点を画面に打っています．

ここで始点が決まったら，再び“／”のデータを読み，FG＝－１となるまで，直線をつないでいくわけですから，FGは0にしておかなければなりません．FG＝0ならば，330行の

LINE −(X,Y),C

という命令で，次々と点をつなげていくことになります．

List 3

```
100 ' List   3
110 '
120 XB=10  : YB=0
130 XE=230 : YE=170
140 XS=(XE-XB)/5 : YS=(YE-YB)/5
150 BC=4 ……バックグラウンドカラー
160 '
170 SCREEN 2 : COLOR 1,BC,BC : CLS
180 FOR I=1 TO 5
190   X=XE-XS*I : Y=YB+YS*(I-1)
200   GOSUB 240
210 NEXT I
220 '
230 DM$=INPUT$(1):END
240 '
250 RESTORE 360
260 READ A$
270   IF A$="@" THEN RETURN
280   IF A$="/" THEN FG=-1 : GOTO 260
290   READ B$,C$
300   DX=VAL(A$) : DY=VAL(B$)
310   C=VAL(C$)   ……文字データを数値に変換する
320   IF FG=-1 THEN PSET (DX+X,DY+Y),C : FG=0 : GOTO 260
330   LINE -(DX+X,DY+Y),C
340 GOTO 260
350 '
360 DATA /,35,9,6, 45,2,6, 51,0,6, 47,8,6, 49,13,6, 48,19,6
370 DATA 43,20,7, 53,28,7, 46,29,7, 36,27,7, 15,24,7, 2,24,7
380 DATA 0,22,10, 1,19,10, 8,15,10, 9,13,10, 23,11,10, 24,9,10,
26,5,10
390 DATA 29,4,2, 35,9,2, /,23,11,2, 35,9,2, /,15,24,2, 43,19,2,
@
```

リンゴを描く

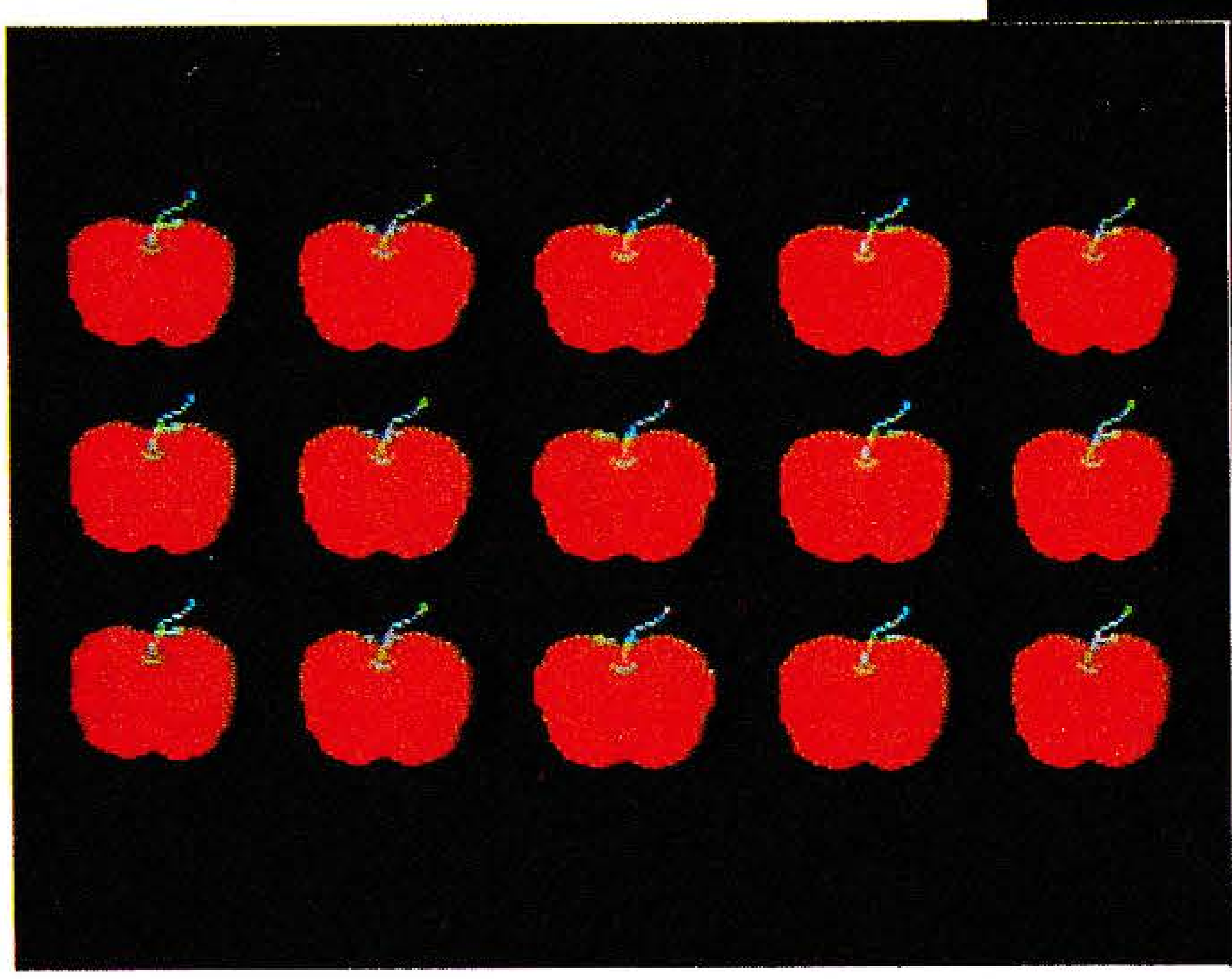
このあたりでいよいよDRAW命令の登場です．このコマンドは，LINE，CIRCLE命令などにくらべて複雑でわかりにくいかもしれませんが，一度理解してしまうととても便利ですので積極的に利用してください．

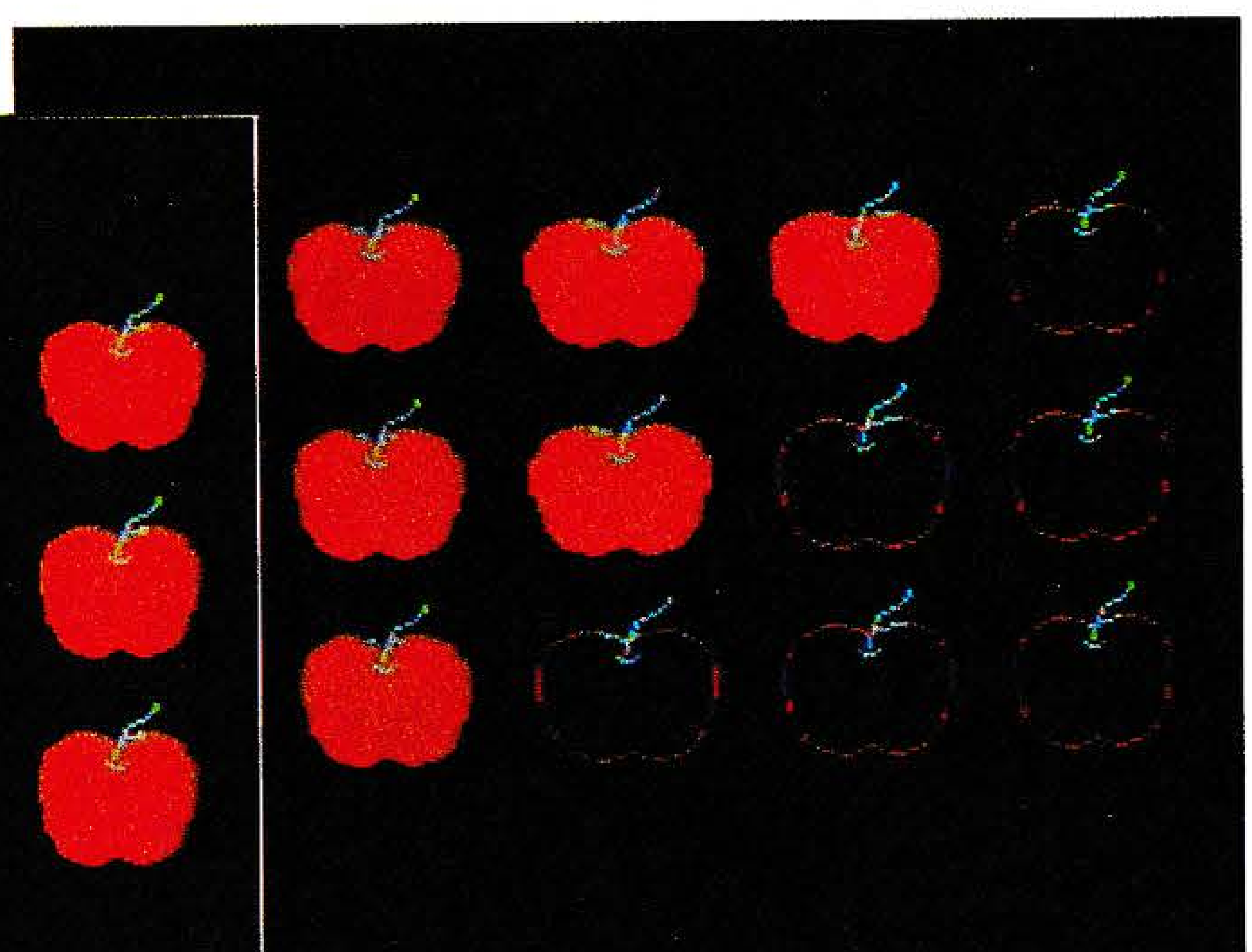
さてこのプログラムですが，“MUG CUPS”や“スペースシャトル”と基本的には同じものです．

List 4 を見てください．170〜210行の2重ループは，リンゴを描くために原点を決めています．このメインルーチンは，“MUG CUPS”と同じものです．240行以降がDRAW命令によりリンゴを描くサブルーチンです．

このサブルーチンを“MUG CUPS”と同じようにプログラムしたものが *List 4a* です．DRAW命令を用いるとプログラムがいかにコンパクトになるか，おわかりでしょう．

話を *List 4* に戻します．このプログラムでは，DRAW命令の中の移動マクロ命令と呼ばれるB及びMを用いました．たとえば，240行のDRAW“BM=I；，=J；”は，座標（I，J）まで線を描かずに始点を移動させることを意味します．始点をきめたら，250行以降のように移動マクロ命令のMにより線を描きながら始点も移動させます．このMコマンドで次々と直線をつなげていくことにより図形を作っていくのです．なお，描かれる線の色は，色指定マクロ命令（Cコマンド）により指定します．





List 4

```
100 ' List   4
110 '
120 XB=0   : YB=0
130 XE=200 : YE=140
140 XS=50  : YS=50
150 '
160 SCREEN 2 : COLOR 7,1,1 : CLS
170 FOR J=YB TO YE STEP YS
180   FOR I=XB TO XE STEP XS
190     GOSUB 240
200   NEXT I
210 NEXT J
220 DM$=INPUT$(1):END
230 '
240 DRAW "S4A0 BM=I;,=J;BM+25,+29"
250 DRAW "C8M+6,-2M+4,+0M+5,+3M+3,+5M+0,+7M-2,+8M-3,+5M-3,+2"
260 DRAW "M-5,+1M-5,-2M-5,+2M-5,-1M-3,-2M-3,-5M-2,-8"
270 DRAW "M+0,-7M+3,-5M+5,-3M+4,+0M+6,+2"
280 PAINT STEP(0,+1),8
290 DRAW "BM+9,-10C15M+0,+1M-2,+2M-4,+2M-2,+4M-1,+4"
300 DRAW "BM-2,+1M+2,+1M+2,-1"
310 RETURN
```

（140行）……表示するリンゴとリンゴの間隔

（160～220行）メインルーチン

List 4a

```
240 RESTORE 340
250 READ A$
260   IF A$="@" THEN RETURN
270   IF A$="/" THEN FG=-1 : GOTO 250
280   IF A$="^" THEN READ X,Y,C : PAINT (X+I,Y+J),C : GOTO 250
290   READ B$,C$ : X=VAL(A$) : Y=VAL(B$) : C=VAL(C$)
300   IF FG=-1 THEN PSET (X+I,Y+J),C :  FG=0 : GOTO 250
310   LINE -(X+I,Y+J),C
320 GOTO 250
330 '
340 DATA /,25,29,8, 31,27,8, 35,27,8, 40,30,8, 43,35,8, 43,42,8
350 DATA 41,50,8, 38,55,8, 35,57,8, 30,58,8, 25,56,8, 20,58,8
360 DATA 15,57,8, 12,55,8,  9,50,8,  7,42,8,  7,35,8, 10,30,8
370 DATA 15,27,8, 19,27,8, 25,29,8
380 DATA ^, 30,30,8
390 DATA /,34,19,15, 34,20,15, 32,22,15, 28,24,15, 26,28,15
400 DATA 25,32,15, 23,33,15, 25,32,15, @
```

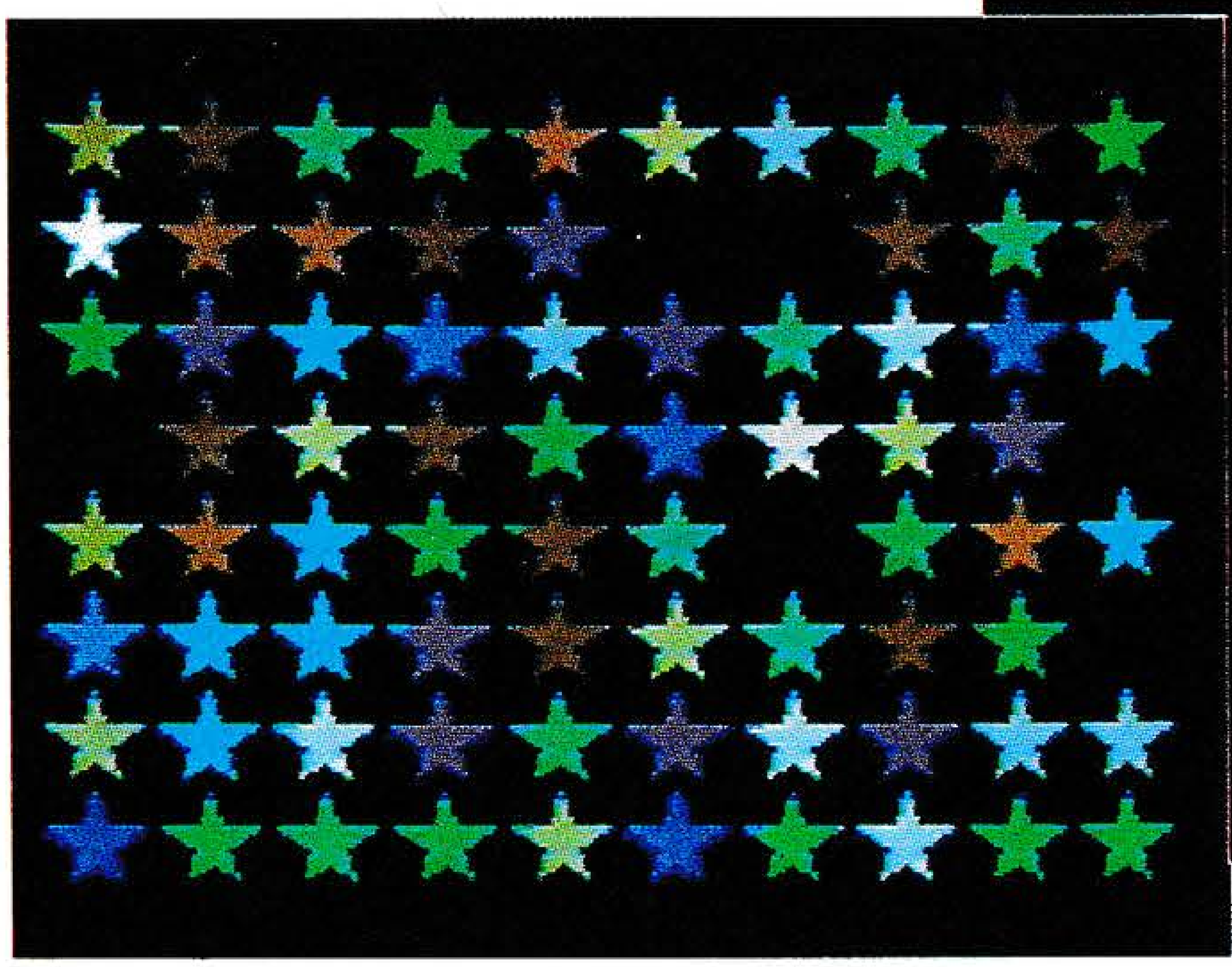
落葉と星型

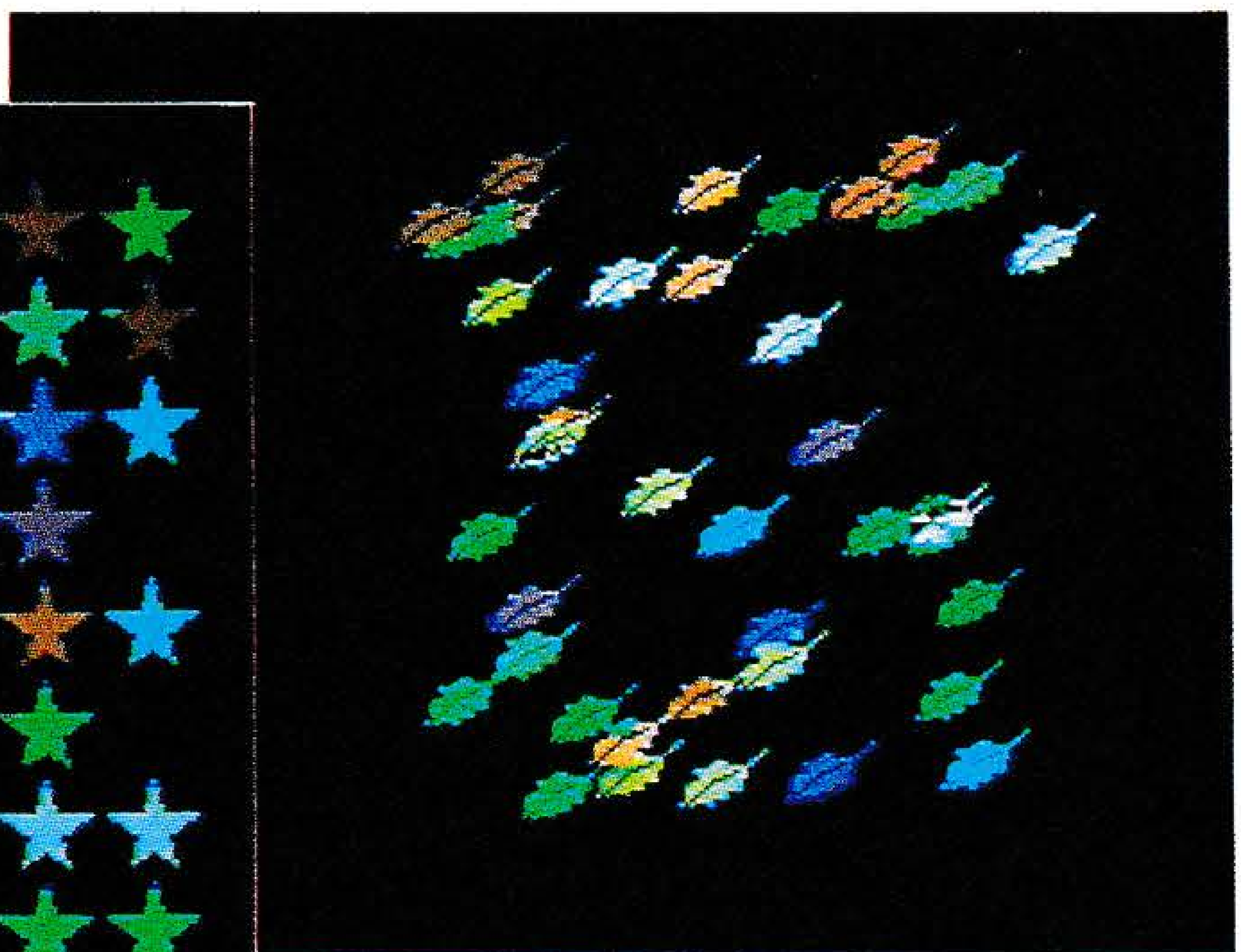
RND（1）という関数をグラフィックに利用してみましょう．いうまでもありませんが，乱数は何ら一定の規則的な並びを持たない数です．それを利用したのが次のプログラムです．

List 5a は，落葉を描くプログラムですが，250行以降のサブルーチンは，（X，Y）を原点として，落葉の形を出力するためのものです．この原点（X，Y）は乱数で決めています．180行では，XB より大きく，XB＋XW より小さい数になるような乱数を発生させて，変数 X に代入しています．同様に190行でも，YB より大きく，YB＋YW より小さくなるような乱数を発生させ，変数 Y に代入しています．200行では，2〜15の乱数を発生させ，変数 C に代入して落葉の色を決定しています．

List 5b は，星型を描くプログラムです．*List 5a* と同じ考え方で，240行以降は，星型を画面に出力するサブルーチンになっています．変数 X には，170，180行で，XS／2から（XD＊XS）＋XS／2までの数を XS おきに発生させ，変数 Y には，0からYD＊YS までの数を YS おきに発生させて代入しています．つまり，得られる数が，おのおの XS，YS おきの乱数となっており，そのため星型が重なり合うことはありません．

このList 5b では XS，YS の大きさを小さくしてみて，画面がどのようになるかためしてみてください．





List 5a

```
100 ' List   5a
110 '
120 XE=190 : YE=170
130 XB=50  : YB=10
140 XW=XE-XB : YW=YE-YB
150 '
160 SCREEN 2 : COLOR 15,1,1 : CLS
170 FOR I=1 TO 40
180   X=INT(RND(1)*XW)+XB
190   Y=INT(RND(1)*YW)+YB      } 乱数による数値の発生
200   C=INT(RND(1)*14)+2
210   GOSUB 250
220 NEXT I
230 DM$=INPUT$(1):END
240 '
250 DRAW "S4A0BM=X;,=Y;BM+20,+0"
260 DRAW "C=C;M-4,+4M-6,-1M+1,+1M-5,+1M+2,+1M-4,+3M+0,+3"
270 DRAW "M-1,+1M+5,-1M+1,+1M+1,-3M+2,+1M+1,-3M+2,+1M+1,-5"
280 PAINT STEP(-7,+4),C
290 DRAW "BM+5,-3C1M-4,+2M-6,+6"
300 RETURN
```

List 5b

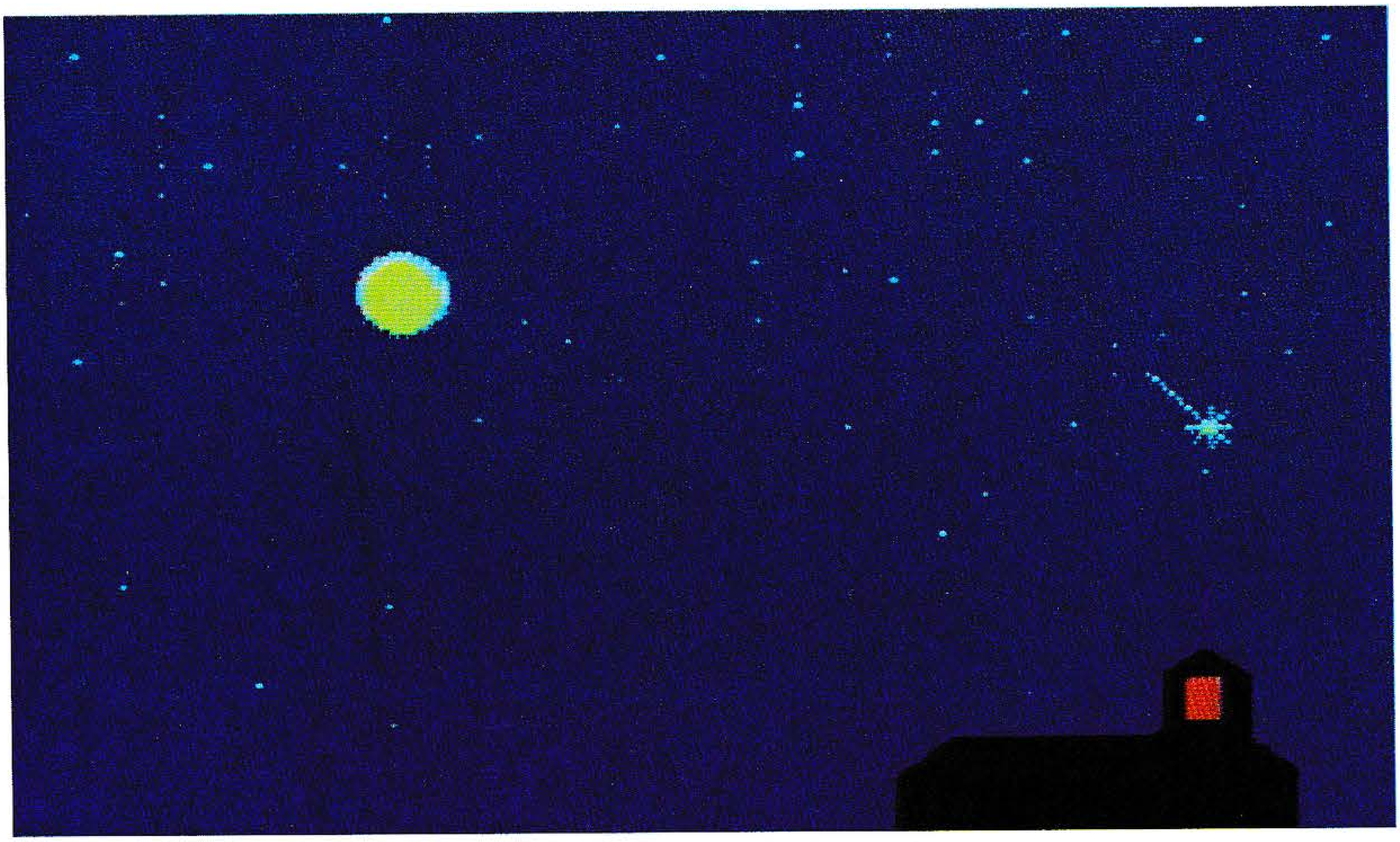
```
100 ' List   5b
110 '
120 XD=10 : XS=24
130 YD=8  : YS=24
140 '
150 SCREEN 2 : COLOR 15,1,1 : CLS
160 FOR I=1 TO 500
170   X=INT(RND(1)*XD)*XS+XS/2
180   Y=INT(RND(1)*YD)*YS          } 乱数による数値の発生
190   C=INT(RND(1)*14)+2
200   GOSUB 240
210 NEXT I
220 DM$=INPUT$(1):END
230 '
240 DRAW "S4A0BM=X;,=Y;BM+10,+0"
250 DRAW "C=C;M-2,+8M-8,+0M+7,+5M-2,+7M+5,-4M+5,+4M-2,-7"
260 DRAW "M+7,-5M-8,+0M-2,-8": PAINT STEP(0,+10),C
270 RETURN
```

Stardust Night

MSX のグラフィック画面は，256×192ドットの解像度があり，見ごたえのある美しい絵を描くことも十分可能です．ここではグラフィックのサンプル・プログラムとしてはスタンダードな，スターダストナイトに挑戦してみましょう．

ここで，プログラムを見てください．まず，バックに散りばめられる星は270～320行で表示しています．280，290行で乱数を発生させ，この位置に星を表示します．ただし，星の分布をＹ座標の小さい方，つまり画面の上の方へ寄せるため，星のＹ座標は290行のように，RND(1)＊RND(1)による乱数を用います．

次に流星ですが，これは400～490行のサブルーチンで描いています．このサブルーチンは，370行のRND(1)<.002という条件が満たされたときにのみ呼び出し実行されます．要するに370行を実行するたびに，星を描くと流れ星の大安売りとなってしまうのを防いだのです．派手好きな人，さびしがりやの人は，370行を，ただのGOSUB 400とし，実行してください．流星自体は乱数で始点，終点を決め，PUT SPRITE命令で描いています．



List 6

```
100 ' List   6
110 '
120 RESTORE 510 : ST$=""
130 FOR I=1 TO 32
140   READ D : ST$=ST$+CHR$(D)
150 NEXT I
```

星のスプライトパターンを文字列として定義する

```
160 '
170 SCREEN 2,2 : COLOR 15,4,1 : CLS
180 DRAW "S4A0 BM160,191 C1"
190 DRAW "M160,178 M170,170 M207,170"
200 DRAW "M207,158 M216,152 M224,158"
210 DRAW "M224,170 M234,178 M234,191"
220 DRAW "BM216,152 M216,140"
230 DRAW "BM212,146 M220,146"
240 PAINT (216,160),1
250 LINE (212,160)-(218,168),8,BF
```

教会を描く

```
260 '
270 FOR I=1 TO 200
280   X=INT(RND(1)*32)*8
290   Y=INT(RND(1)*RND(1)*96)*2
300   C=INT(RND(1)*14)+2
310   PSET (X,Y),C
320 NEXT I
```

星を描く

```
330 '
340 CIRCLE (75,80),8,11 : PAINT (75,80),11
```

……月を描く

```
350 '
360 SPRITE$(0)=ST$
370 IF RND(1)<.002 THEN GOSUB 400
```

……流星を描くサブルーチンを呼ぶ

```
380 IF INKEY$<>"" THEN END ELSE 370
390 '
400 X0=INT(RND(1)*100)+20  : Y0=INT(RND(1)*30) : Y=Y0
410 X1=INT(RND(1)*100)+130 : Y1=INT(RND(1)*70)+60
420 YS=(Y1-Y0)/(X1-X0)*4
430 PLAY "T158L64N80N79N78N77N76N75N74"
440 FOR X=X0 TO X1 STEP 4
450   PUT SPRITE 0,(X,Y),11,0
460   Y=Y+YS
470 NEXT X
480 PUT SPRITE 0,(X,209),11,0
490 RETURN
```

星を動かすサブルーチン

```
500 '
510 DATA 128,64,32,16,8,4,2,1
520 DATA 0,0,0,1,0,0,0,0
530 DATA 0,0,0,0,0,0,0,16
540 DATA 146,84,56,255,56,84,146,16
```

流星のスプライトパターン・データ

円は異なもの味なもの

円を基本とした模様を描く際ひんぱんに登場するのが，例のサインやコサインという三角関数です．

Figure 1 のような直角三角形において，角Ｔと向かい合う位置の辺の長さが sin(T) に相当し，角Ｔと直角を結ぶ辺の長さが cos(T)に相当する，この位置関係を三角形がどんな向きに傾いても見分ける眼力は必要です．

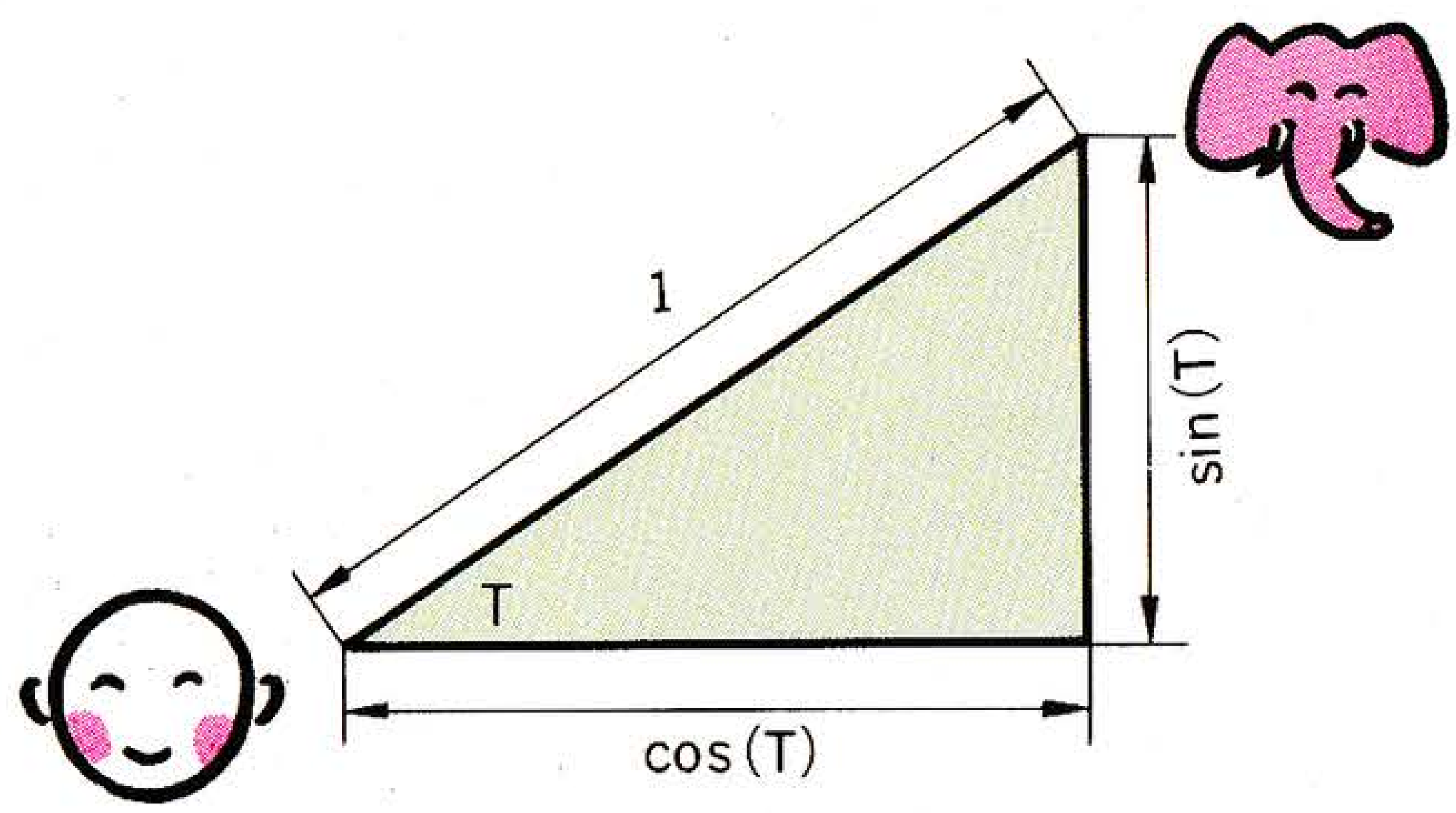
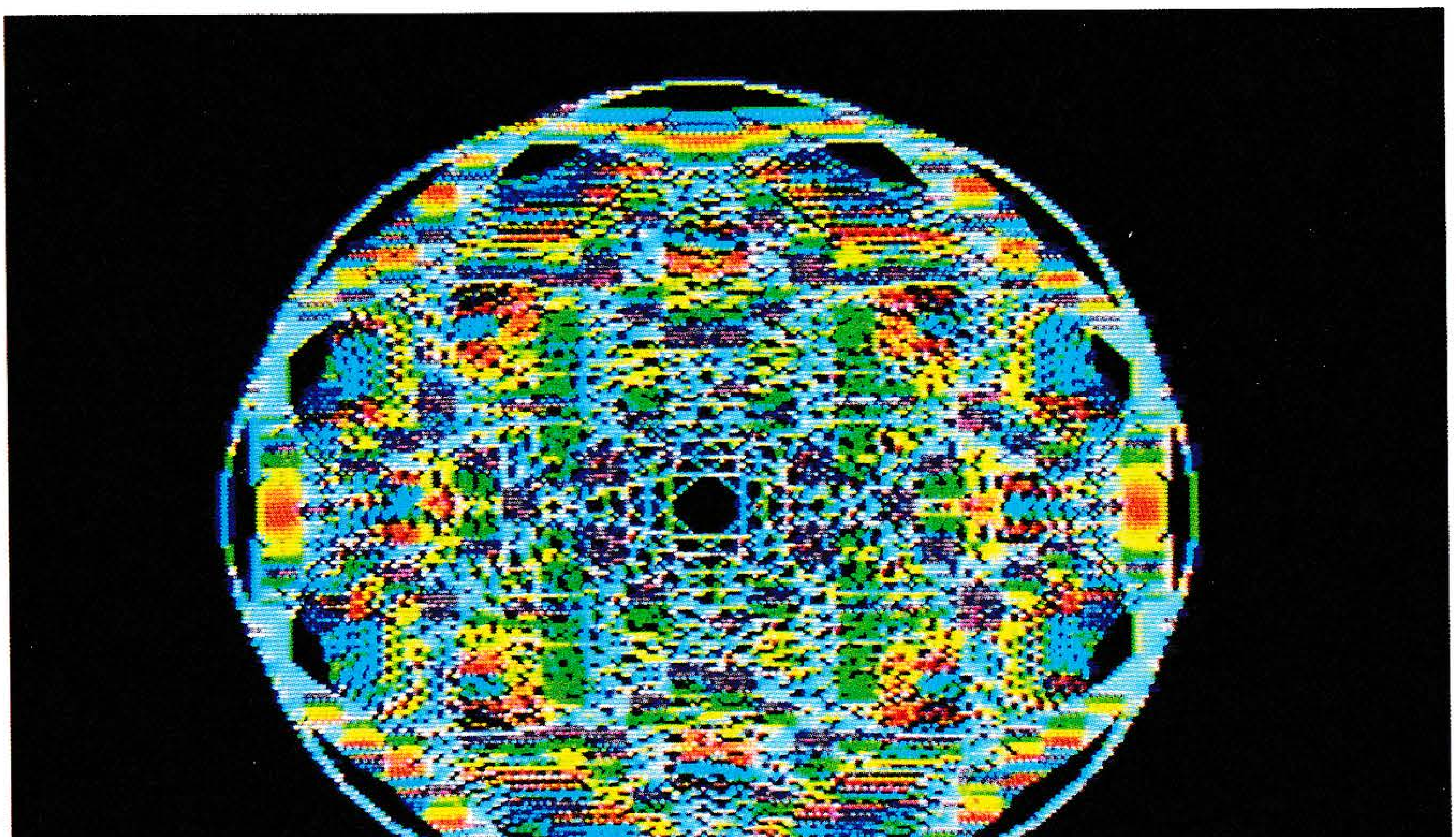


Figure 1



ここで *Figure 2* を見てください．始点から角度 T だけ回転した円周上の位置は，sin関数 と cos関数 を使って計算できます．この位置は，始点の位置と角 T の向きによって様々な場合がありますが，ここでは2通りの例を示しました．どちらの場合も，画面上の x 座標と y 座標はそれぞれ右と下へ行くほど増加するということと *Figure 1* の関係を考え合わせれば，答が出てきます．

円の中心を(X0，Y0)とすれば，*Figure 2a* のように，円の右端から反時計回りに角Tだけ回転した位置は(X0+cos(T)，Y0−sin(T))，また，*Figure 2b* のように，円の下の端から時計回りに回転させる場合は (X0−sin (T)，Y0+cos (T))というように位置の計算ができるわけです．

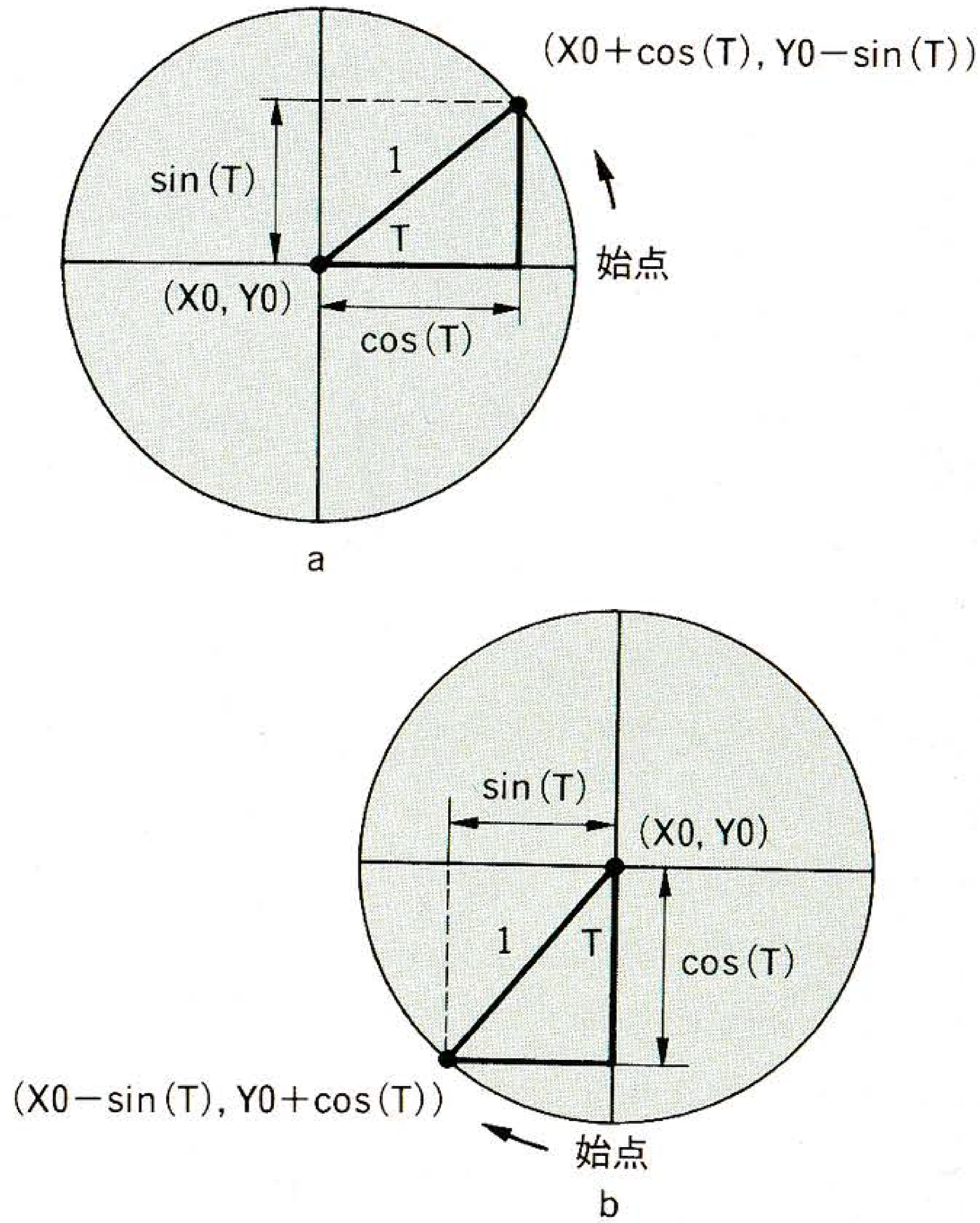
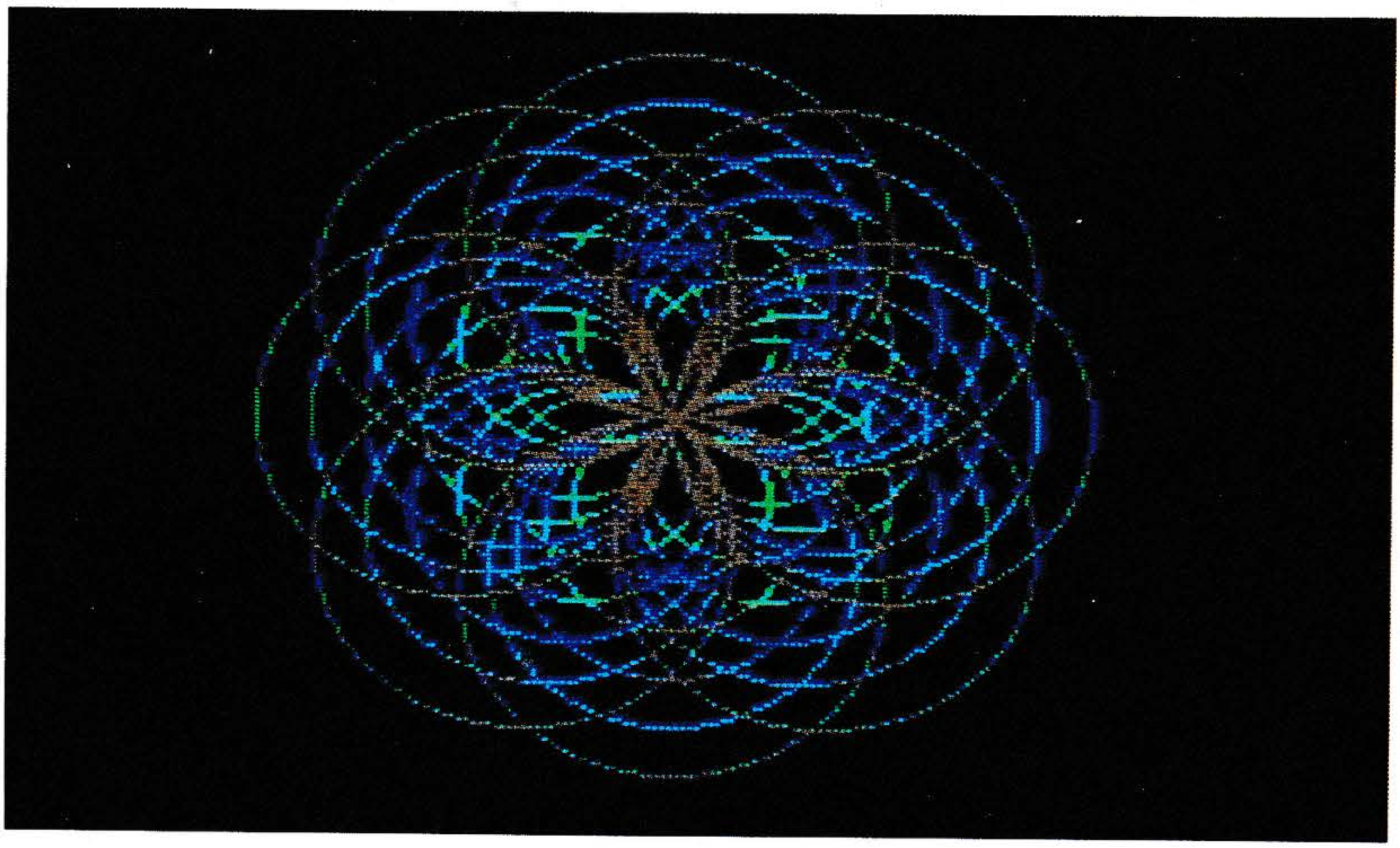


Figure 2



List 7 では，上の例の *Figure 2a* と同じ角度のとり方を用いて，半径 R の円周上に次々と小さな円を描き並べます．そこで200行のようにこの円を描く座標には，半径 R を掛け，(X0 + cos (T) * R，Y0 − sin (T) * R) としなければなりません(*Figure* 3 参照)．また，180〜240行の FOR 〜 NEXT ループでRの値を RB から RE まで次第に大きくしながら，同様なことを繰り返していきます．

さて，今のはほんの小手調べです．次に210行を *List 7a* のように変更し，どんな図形が現れるか見てください．これは R が大きくなるにつれ，描かれる円も大きくなっています．

では，140，210行を *List 7b* のように変えてみましょう．今度は，半径96の大きな円周の内側に接して，円が並びます．

このように，円の半径の指定を変えると，いろいろな変化が楽しめます．さらに，ここで SCREEN 3のマルチカラーモードを使ってみました．*List 7c* を追加してみてください．

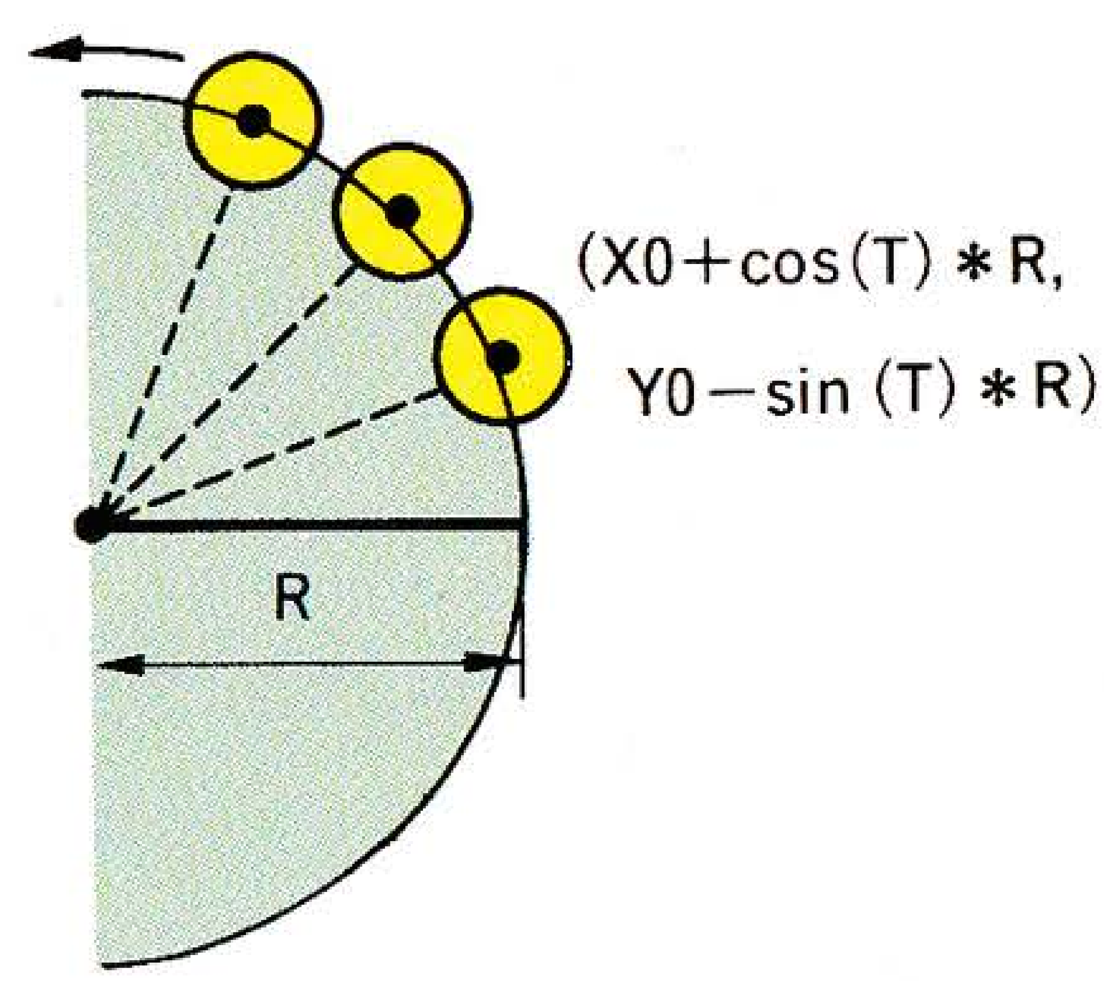
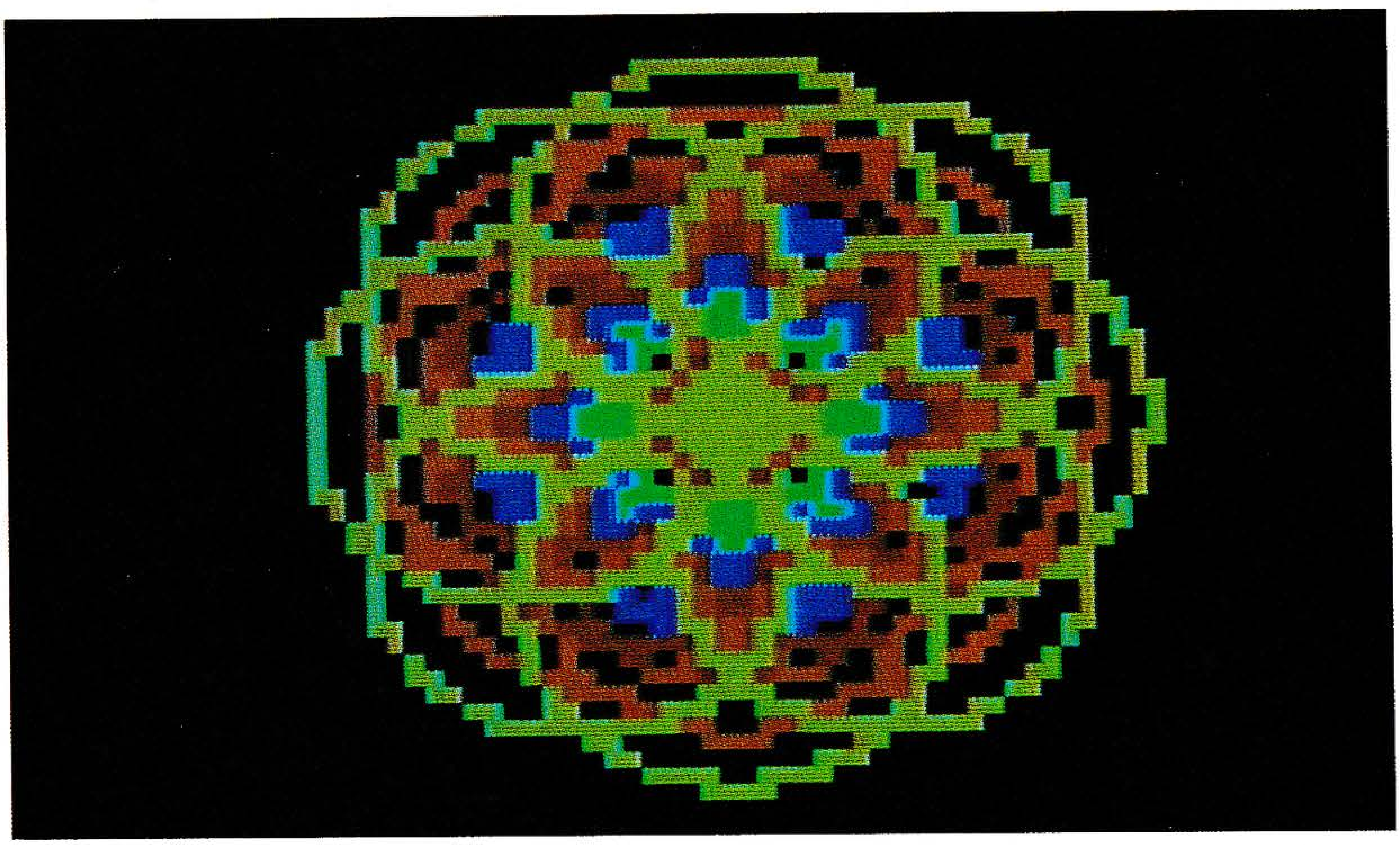


Figure 3



List 7

```
100 ' List   7
110 '
120 DEFSNG A-Z
130 PI=3.14159 ………定数の設定
140 RB=20 : RE=44 : RS=6
150 S=8 : C=2
160 '  ………円周を8分割する
170 SCREEN 2 : COLOR 14,1,1 : CLS
180 FOR R=RB TO RE STEP RS
190   FOR T=0 TO 2*PI STEP 2*PI/S
200     X=128+R*COS(T) : Y=96-R*SIN(T)
210     CIRCLE (X,Y),4,C
220   NEXT T
230   C=C+1 : IF C=16 THEN C=2
240 NEXT R
250 DM$=INPUT$(1):END
```

List 7a

```
210     CIRCLE (X,Y),R,C
```

List 7b

```
140 RB=20 : RE=44 : RS=2
210     CIRCLE (X,Y),96-R,C
```

List 7c

```
140 RB=20 : RE=44 : RS=6
150 S=8 : C=2
170 SCREEN 3 : COLOR 14,1,1 : CLS
210     CIRCLE (X,Y),R,C
230   C=C+2 : IF C>16 THEN C=2
```

自分だけの曲線を描こう

sin 関数や cos 関数はグラフィックには欠かせない関数です．ここでは，数学的な知識はいっさいぬきにして，三角関数が周期的に変化するという性質を利用して，いろいろな曲線を作ってみましょう．

さて，*List 8* を見てください．

220～290行の FOR～NEXT ループは，230，240行の sin，cos 関数の引数を，0から2πまで，π／180ずつ変化させています．このπは，円周率を表し，3.14159……という値を持っています．sin 関数，cos 関数ともに，引数を0～2πまで少しずつ変化させていったとき，sin(A)，cos(A) とは，*Figure 1* に示されるような関数なのです．この sin(A)，cos(A) 関数の値は，2πのところで，ちょうどはじめの0の値に戻ります．これが，三角関数の1周期なのです．

変数 A の値を0から2πまで変化させると，0と2πでは，sin(A)，cos(A)が同じ値をとることから，何となく，グラフの書き始めと書き終わりが一致するだろうということに気づきましたか．

250行は，変数 X，Y の値を，画面にちょうど収まる程度の大きさにしている部分です．もしここで値が大きすぎて，画面からはみ出してしまうような場合には，260，

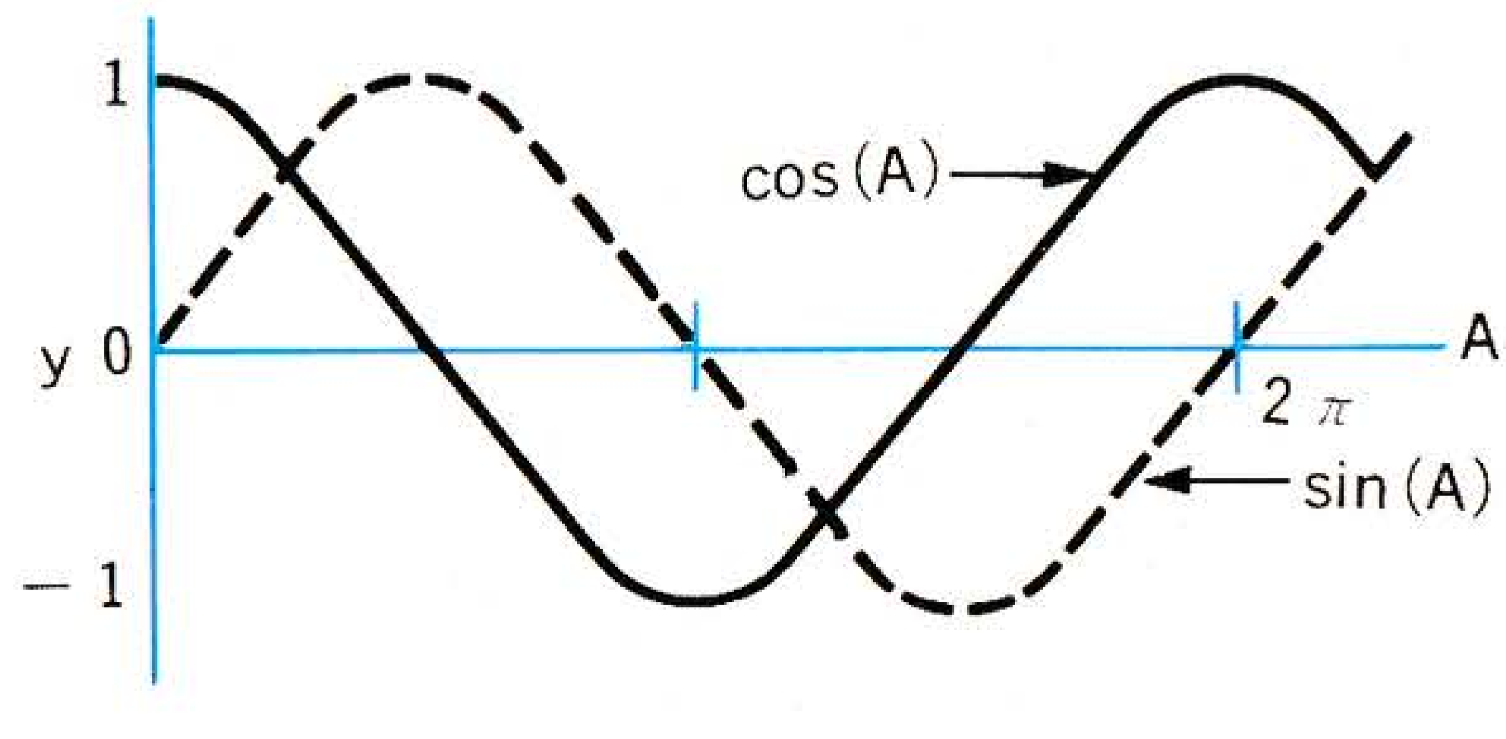
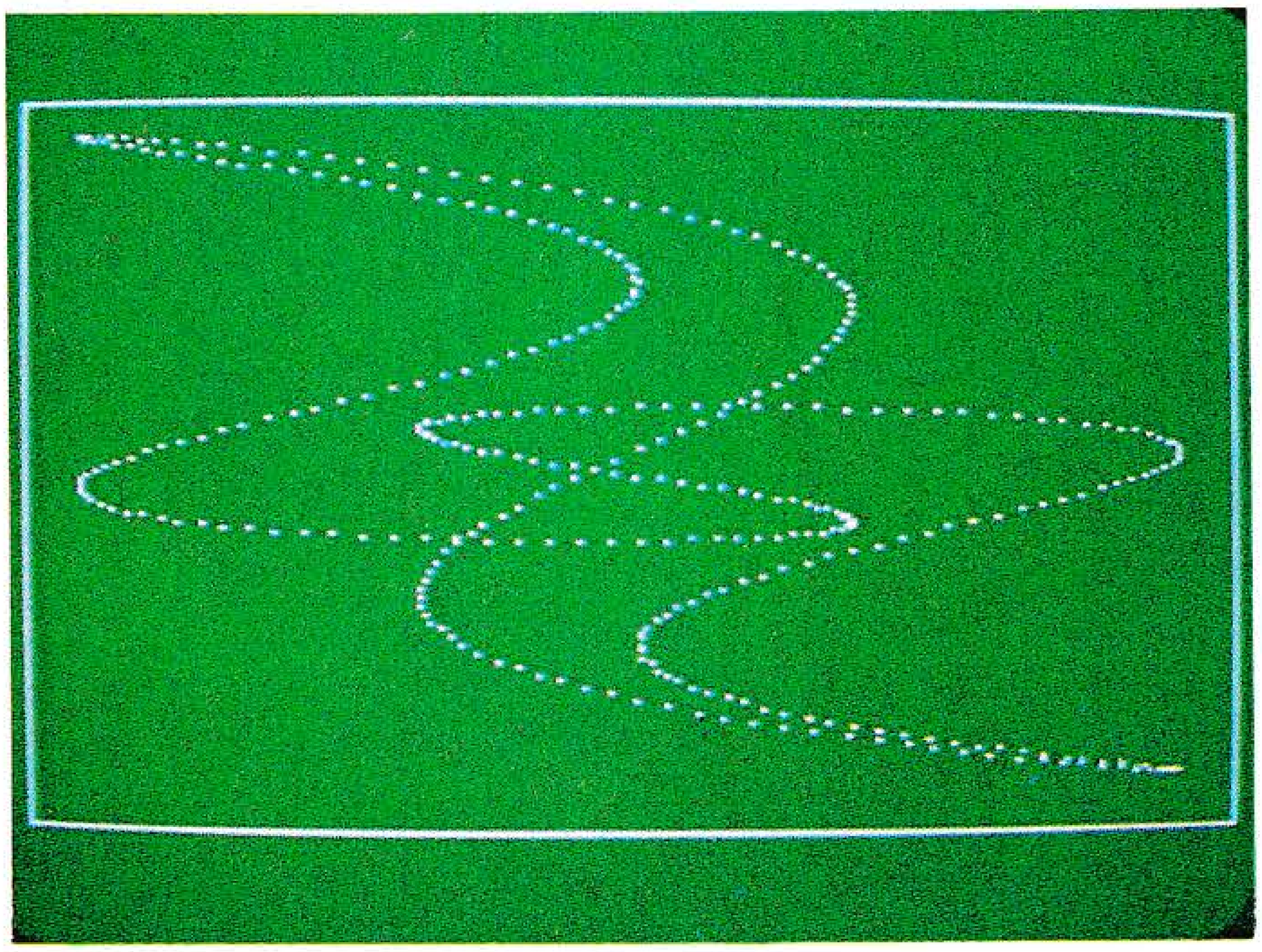
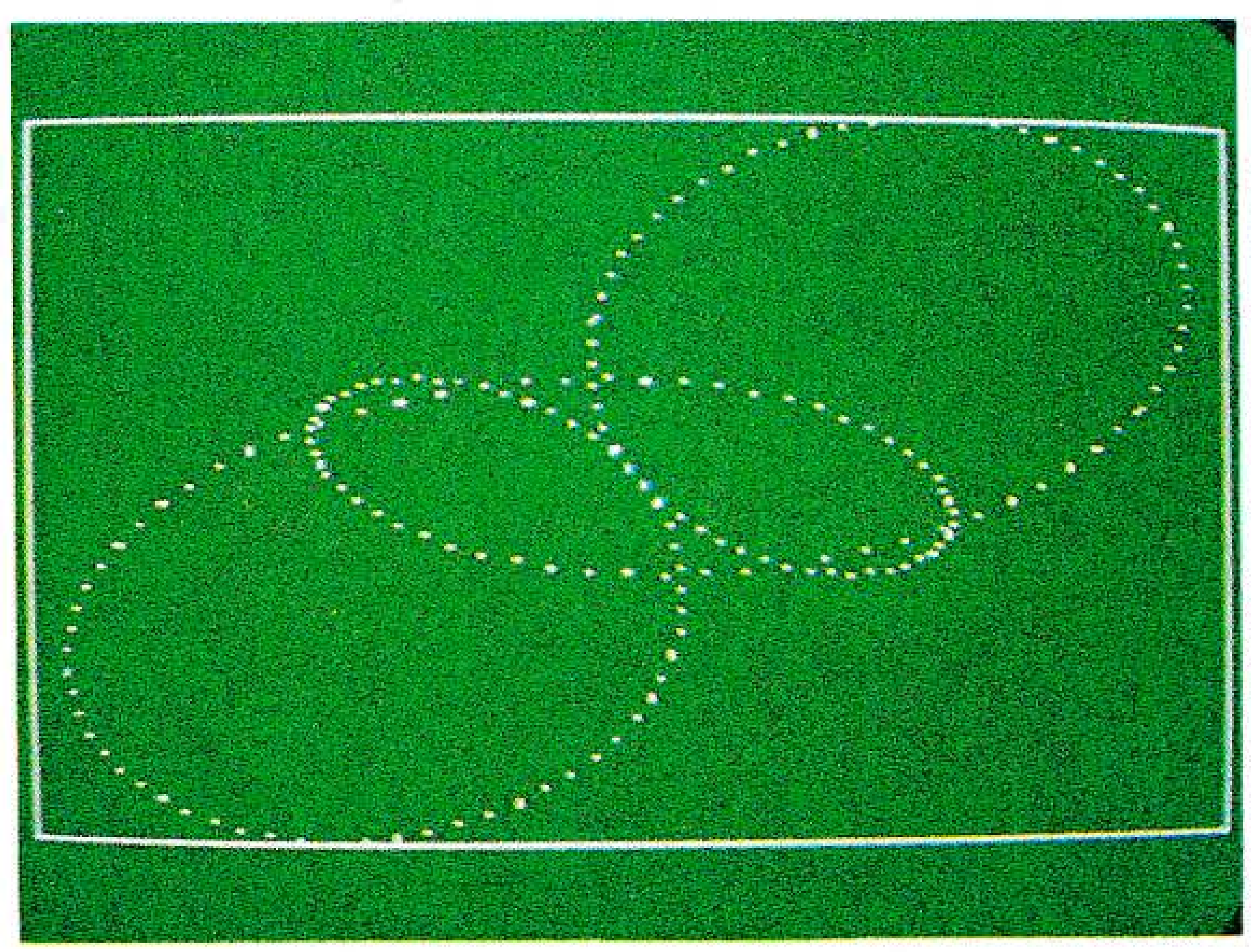


Figure 1





270行でそれを判断し，280行の PSET 命令を飛び越えるようになっています．

自分だけの曲線を作り出すといっても，何の手がかりもなしに行うことはむずかしいでしょう．そこで230行と240行の変数 X と Y をどのように書き換えたらよいのかを，*List 8a〜f* に示しましたのでためしてみてください．

List 8

```
100 ' List   8
110 '
120 DEFSNG A-Z
130 PI=3.14159            ………定数の設定
140 XE=250 : YE=180      } グラフィックを表示する範囲
150 XB=10  : YB=10       }
160 XM=(XE-XB)/2+XB : YM=(YE-YB)/2+YB
170 XW=(XE-XB)/4    : YW=(YE-YB)/4
180 DC=15 : BC=15         ………プロットする点，バックグラウンドの色を白に設定する
190 '
200 SCREEN 2 : COLOR 14,12,12 : CLS
210 LINE (XB,YB)-(XE,YE),BC,B
220 FOR A=0 TO 2*PI STEP PI/180
230   X=COS(A)-COS(A*5)          } プロットする関数
240   Y=SIN(1.5+A)-SIN(A*2)      }
250   PX=X*XW+XM : PY=Y*YW+YM
260   IF PX<XB OR PX>XE THEN 290
270   IF PY<YB OR PY>YE THEN 290
280   PSET (PX,PY),DC
290 NEXT A
300 DM$=INPUT$(1):END
```

List 8a

```
230   X=SIN(A)
240   Y=COS(A)
```

List 8b

```
230   X=SIN(A*4)
240   Y=COS(A)
```

List 8c

```
230   X=SIN(A*4)
240   Y=COS(A*3)
```

List 8d

```
230   X=SIN(A*4)+COS(A)
240   Y=SIN(A)-COS(A)
```

List 8e

```
230   X=SIN(A*2)-COS(A*5)
240   Y=SIN(1.5+A)-SIN(A*6)
```

List 8f

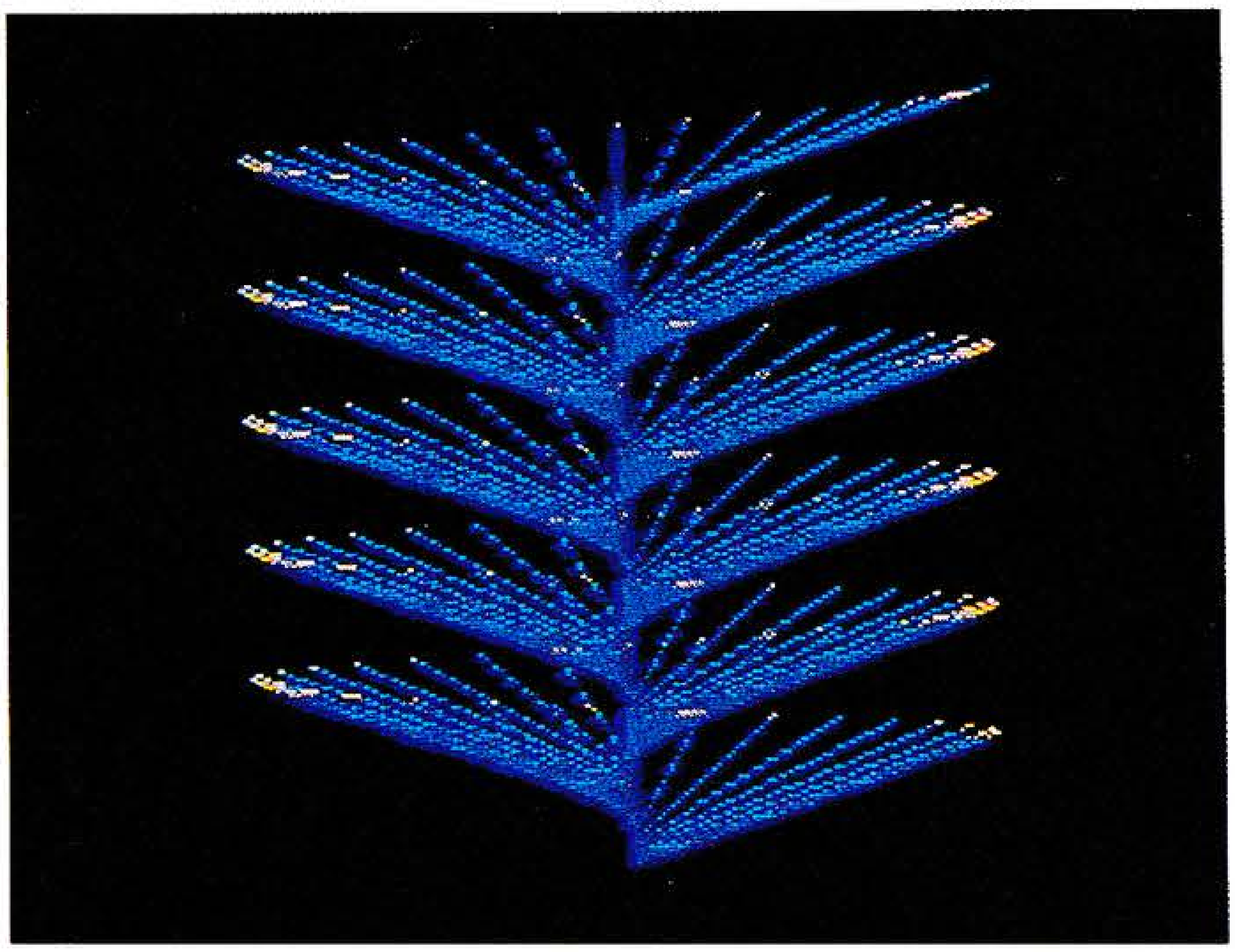
```
230   X=COS(A*6)-SIN(A*2)
240   Y=COS(A*2)-SIN(A*6)
```

ヘリコイド曲線を描く

様々な曲線の中には，その描き方や形から，特定の名前がつけられているものがあります．もちろん，「円」もそうですが，サインカーブ，コサインカーブ，2次曲線，3次曲線，またちょっとむずかしいものでは，リサージュ図形，サイクロイドなんてものもあります．ここでいう，ヘリコイドも，そんな名前のついた曲線のひとつです．

ヘリコイド曲線は，「中心となる軸のまわりを，1点が少しずつ回りながら下へ向かって降りていく様子を，横から眺めたもの」と考えてください（*Figure 1* 参照）．

1点がある軸を中心に回転しながら降りていく場合に，それを横から見ると，*Figure 1b* のようになります．これは単なるコサインカーブです．そうです，実はこれがヘリコイドの秘密なのです．このコサインカーブと中心軸とを一定の間隔をあけて結びつけてやると，1点が軸を回りながら降下していくように見えるのです．*List 9* では，回転しながら降りていく点の Y 座標 YH と，結びつける中心軸上の Y 座標 YL には，YL−YH=30の関係が成り立つようにしました（*Figure 2* 参照）．



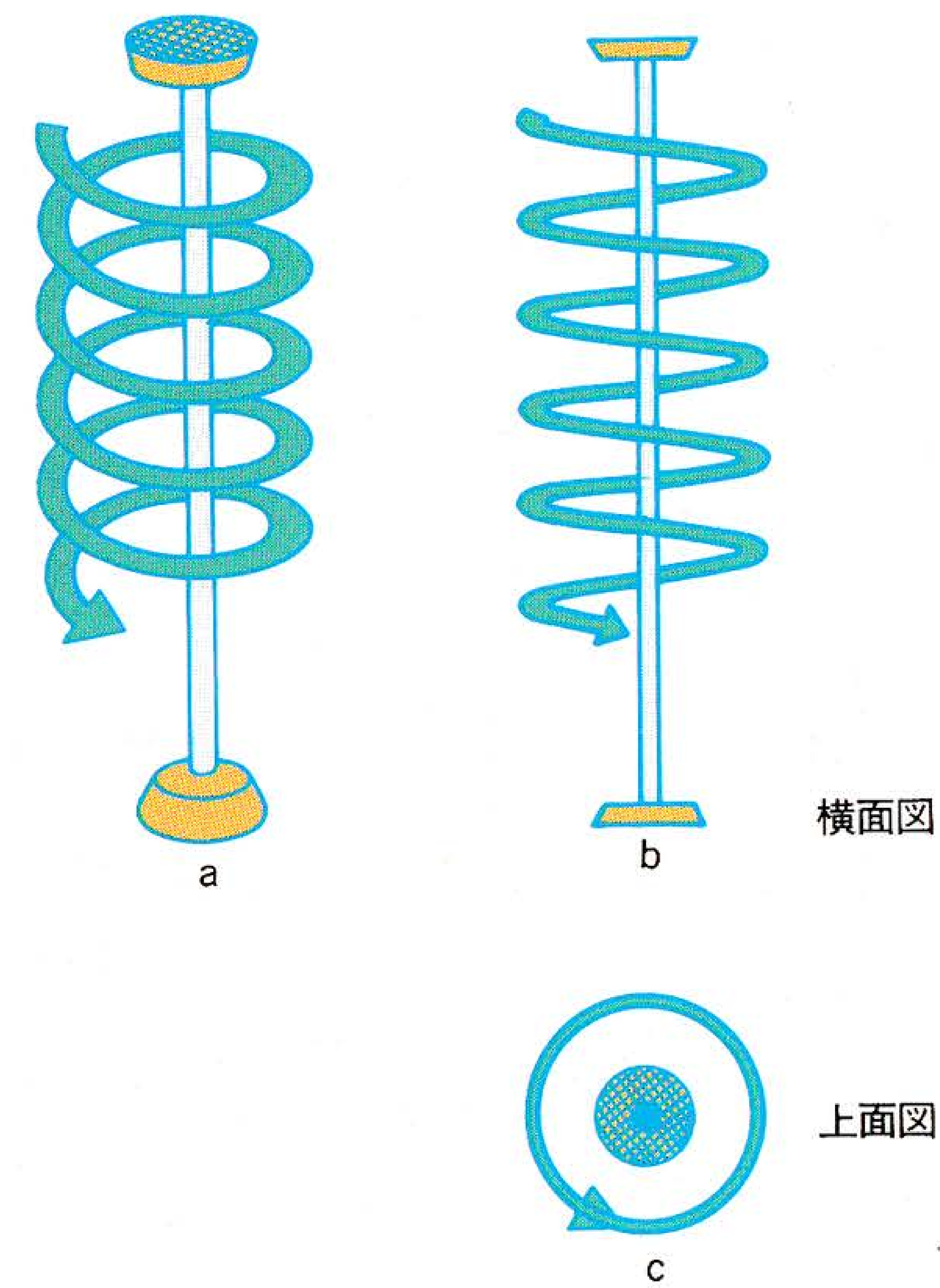


Figure 1

ここで，コサインカーブを描くためには，三角関数のひとつである cos 関数を用います．このcos関数は，その引数を0～2π（π=3.14159……）まで少しずつ変化させると，ちょうど0と2πで同じ値になり，それを繰り返すことは前にも述べたとおりです．

ここでは，190行で T を0～10πまで変化させているので，計5周することになります．ここで，T はπ／16ずつ動かしているので，一周（2π）するまでに，中心の軸と32個の点が結ばれることになります．

このプログラムでは，中心軸と点との距離を変数 R として，それを160～250行のFOR～NEXT ループで，RS おきに変化させています．したがって，画面が進むにつれて，図形の横幅は広くなっていきます．

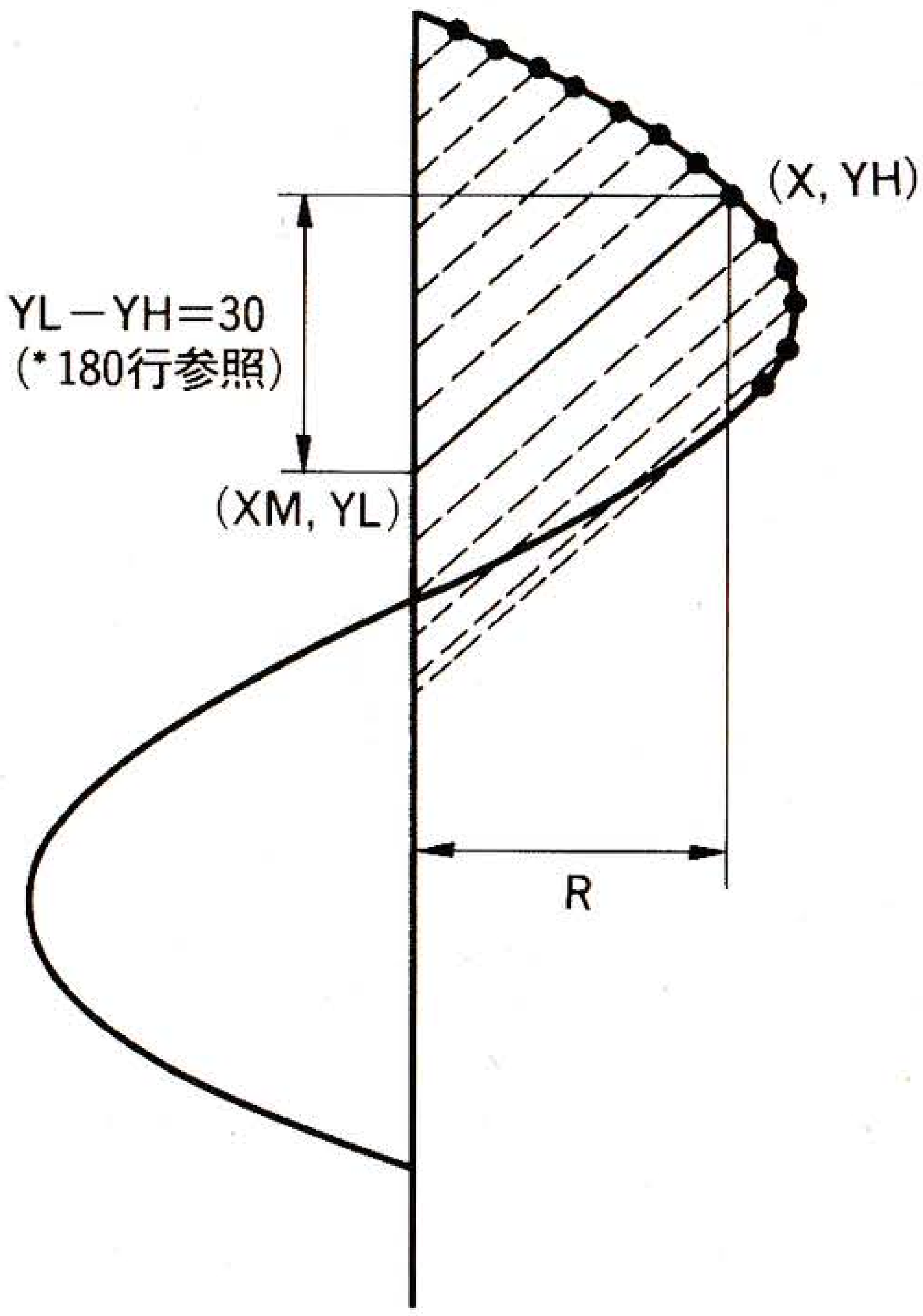


Figure 2

List 9

```
100 ' List   9
110 '
120 PI=3.14159………定数の設定
130 RB=20 : RE=80 : RS=60 : XM=128
140 '
150 SCREEN 2 : COLOR 15,1,1 : CLS
160 FOR R=RB TO RE STEP RS
170   CLS
180   YL=30 : YH=0
190   FOR T=0 TO 10*PI STEP PI/16
200     YL=YL+1 : YH=YH+1
210     X=R*COS(T)+XM
220     LINE (XM,YL)-(X,YH),4 : PSET (X,YH),9
230   NEXT T
240   FOR T=1 TO 500 : NEXT T
250 NEXT R
260 DM$=INPUT$(1):END
```

（190～240行）ヘリコイド曲線を描く

ネコでも回そうか?

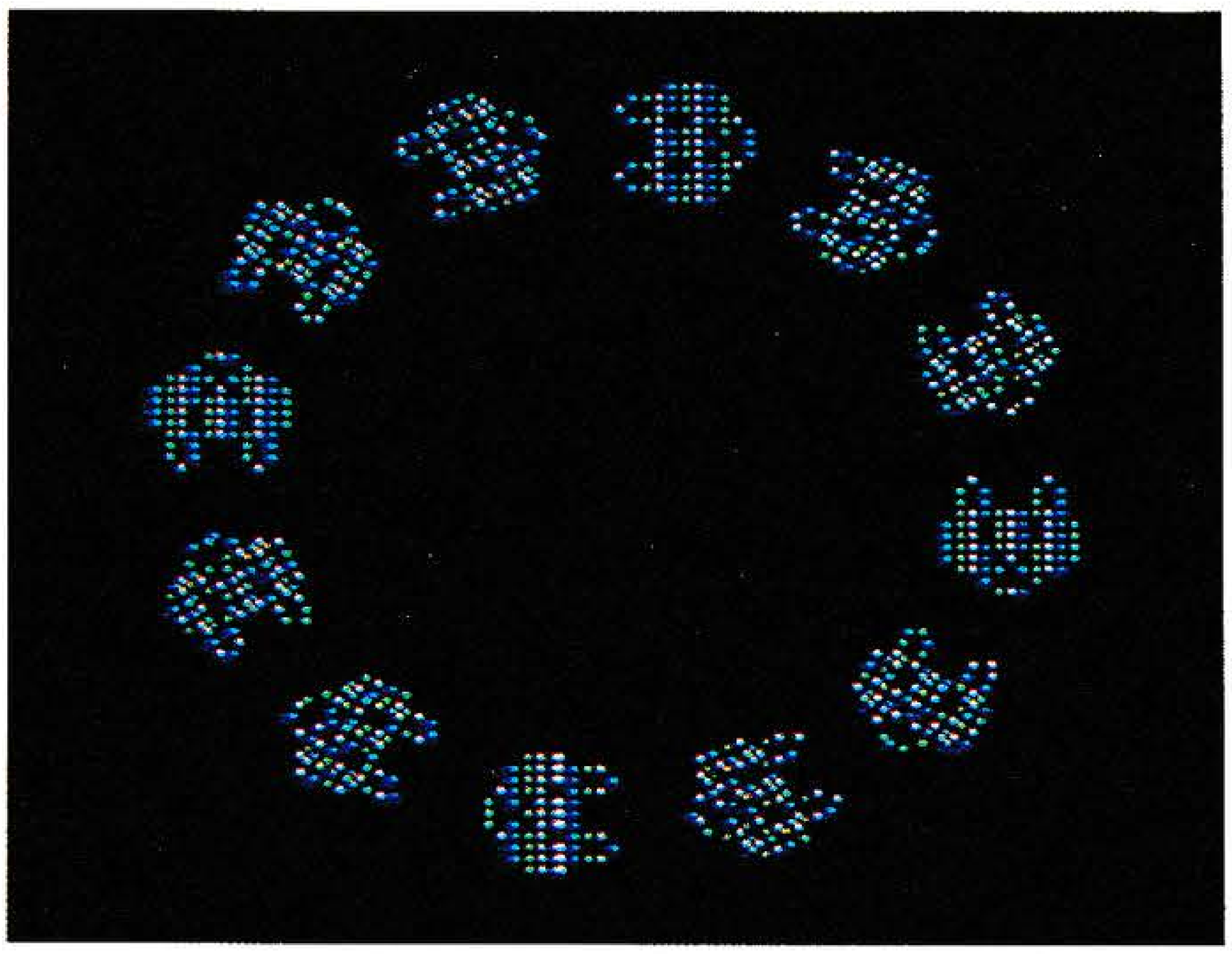
グラフィックの操作の中で，意外にむずかしいのは図形を傾けたり回転させたりすることです．たとえばネコの顔を描くデータがここにあるとします．これを使って少し傾いたネコを表示しようと思ったら，まず顔の右端をちょっと下げて左端は上げ，頭のテッペンは右へずらしてアゴは左へ，というような操作が必要になるわけです．でも，こんなことをいちいち自分で座標の計算を思案してたらたいへんです．そこで，ここでは図形を回転させたときの位置を得るのに便利な式を紹介しましょう．

図形は，基本的には点の集まりです．いま，回転の中心から(X，Y)だけ離れた位置にある点を1つ考えたとき，それが角度Tだけ回転したあとの位置（XT，YT）は，

$$\begin{cases} XT=\cos(T)\times X-\sin(T)\times Y \\ YT=\sin(T)\times X+\cos(T)\times Y \end{cases}$$

という式で算出できます（*Figure 1* 参照）．この式を利用したのがこのプログラムです．

回転させるデータは121個の点から構成されます．これらは回転の中心位置から右へRだけ離れた場所に，x軸方向XS，y軸方向YSの距離を互いに保って並んでいます．この格子の左からI番め，上からJ番



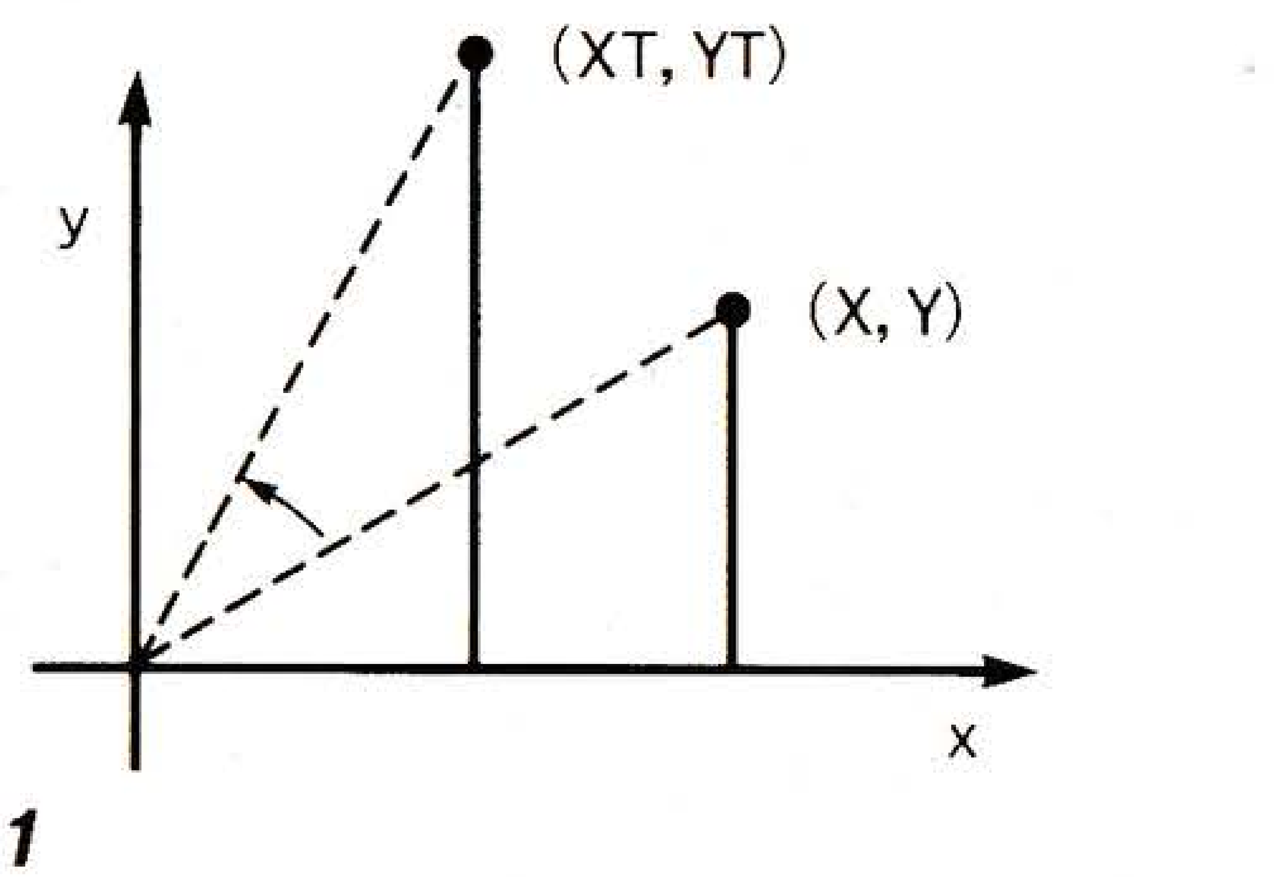


Figure 1

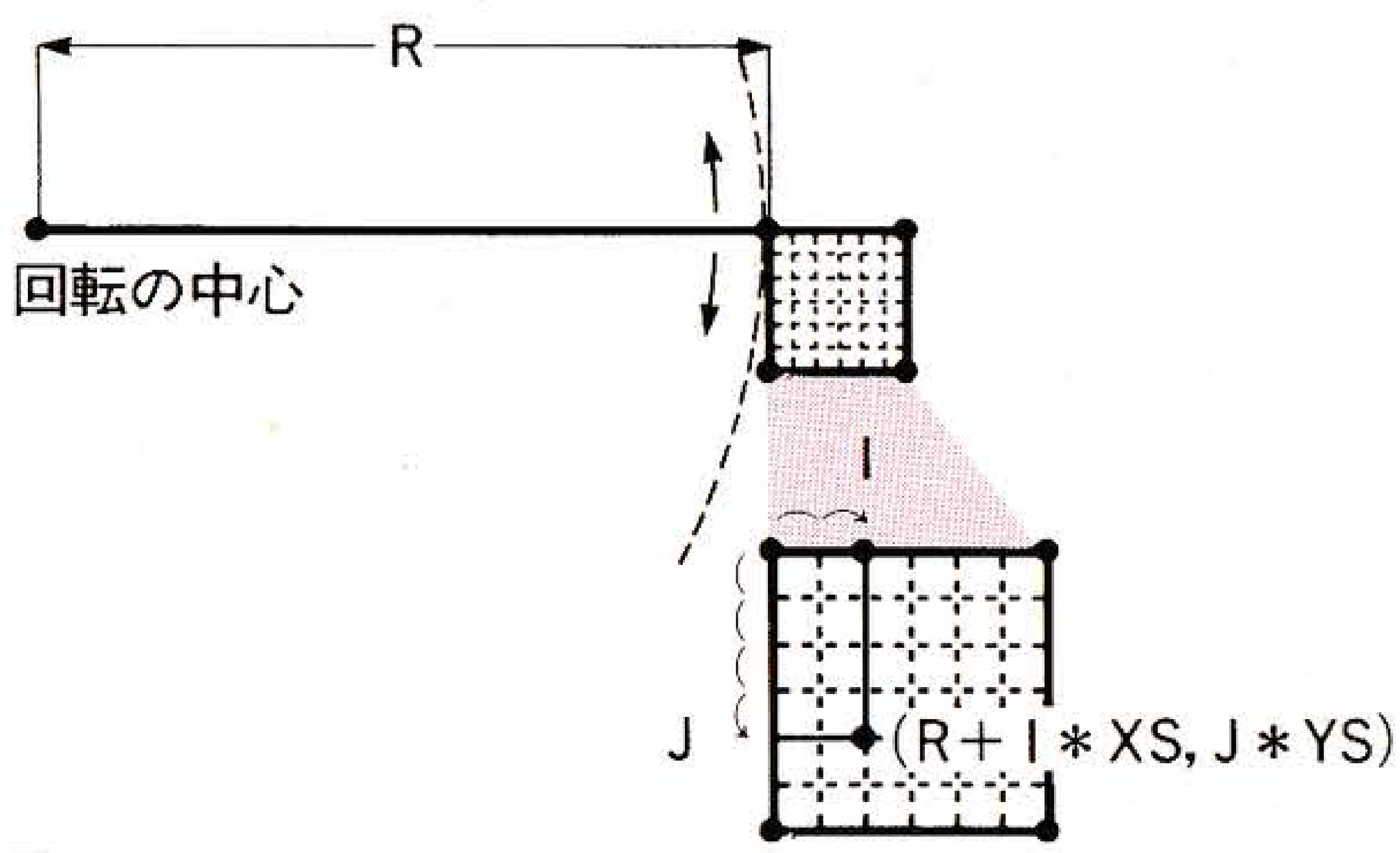


Figure 2

目に位置するドットの座標は，*Figure 2* のように（R+I＊XS，J＊YS）と表されます．さらにこの位置のドットの色は，配列 PC（I，J）に格納しておきます．

プログラムの250〜350行のループでは，角度 T を変えて，各ドットの位置を計算して回転させ，PC(I，J)の色で点を打っています．

List 10

```
100 ' List  10
110 '
120 DEFSNG A-Z
130 DG=3.14159/180    ………定数の設定
140 R=50
150 XS=2 : YS=2
160 D=10 : DIM PC(D,D)
170 '
180 FOR J=0 TO D
190   FOR I=0 TO D
200     READ PC(I,J)     データの読込み
210   NEXT I
220 NEXT J
230 '
240 SCREEN 2 : COLOR 14,1,1 : CLS
250 FOR T=0 TO 359 STEP 30
260   C=COS(T*DG) : S=SIN(T*DG)
270   FOR J=0 TO D
280     FOR I=0 TO D
290       X=R+I*XS : Y=J*YS
300       XT=C*X-S*Y                          メインルーチン
310       YT=S*X+C*Y
320       PSET(128+XT,96+YT),PC(I,J)
330     NEXT I
340   NEXT J
350 NEXT T
360 DM$=INPUT$(1):END
370 '
380 DATA 1,1,9,1,1,1,1,1,9,1,1
390 DATA 1,9,1,9,1,1,1,9,1,9,1
400 DATA 1,9,1,9,1,1,1,9,1,9,1
410 DATA 1,9,9,9,9,9,9,9,9,9,1
420 DATA 9,9,9,1,9,9,9,1,9,9,9
430 DATA 9,9,9,1,9,9,9,1,9,9,9    回転させるネコのデータ
440 DATA 9,9,9,9,9,9,9,9,9,9,9      カラーコード  1…黒
450 DATA 9,9,9,9,1,9,1,9,9,9,9                    9…赤
460 DATA 1,9,9,9,9,1,9,9,9,9,1
470 DATA 1,1,1,9,1,1,1,9,1,1,1
480 DATA 1,1,1,1,9,9,9,1,1,1,1
```

3次元グラフィックス？

最近，TVコマーシャルなどでも，非常に美しい3次元グラフィックスが見られるようになりましたが，その美しさに感激して，パソコンで同様のことを行いたいと思っても，複雑な3次元→2次元変換などが必要ですし，プログラムも膨大なものになってしまい，簡単には手が出せません．しかし，複雑な理論を知らなくても「3次元グラフィックス」は可能です．ここではそのテクニックを紹介します．

このプログラムで行っていることは，基本的には，*Figure 1* のように，表示したい立体をスライスし，それを少しずつずら

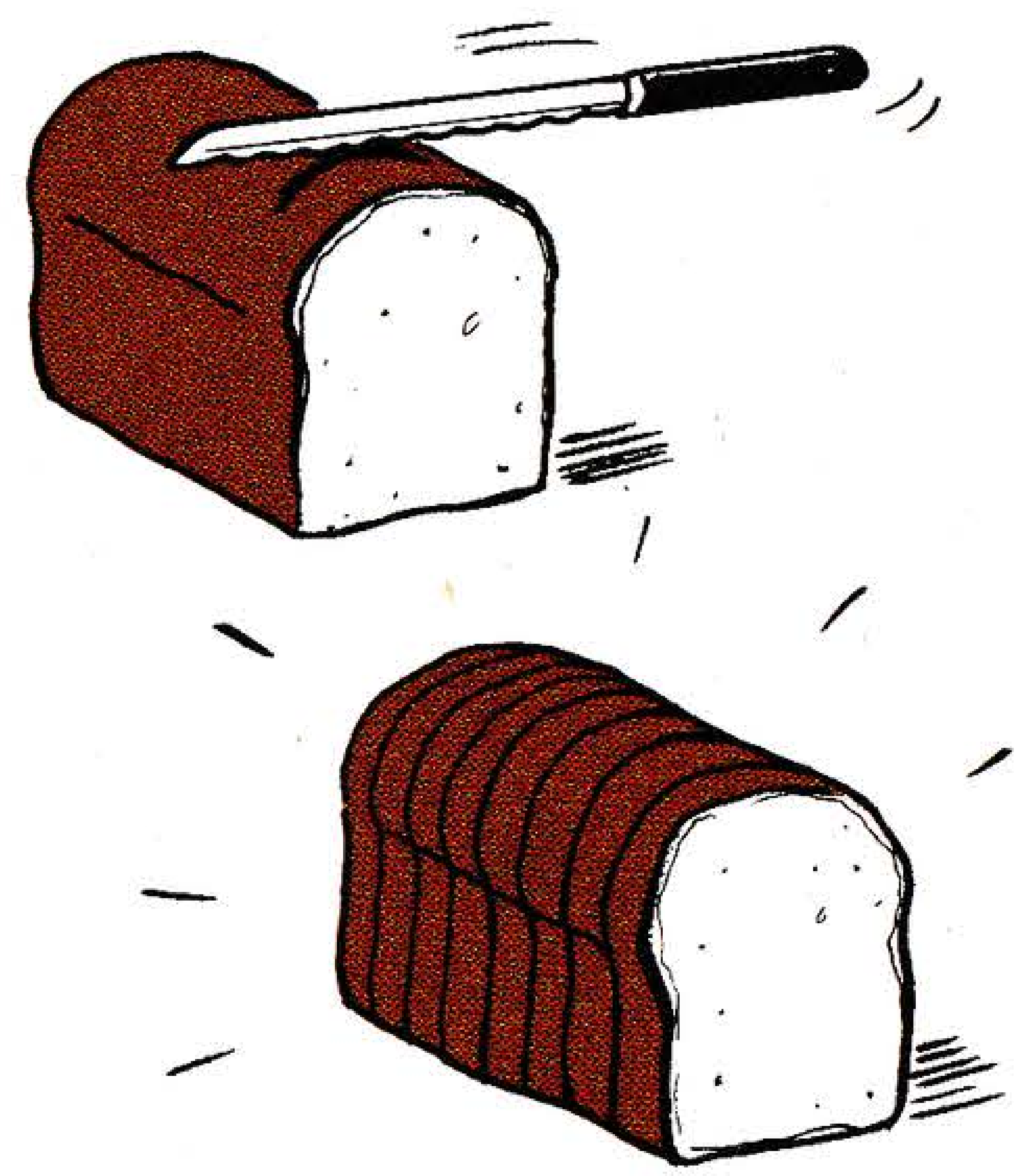
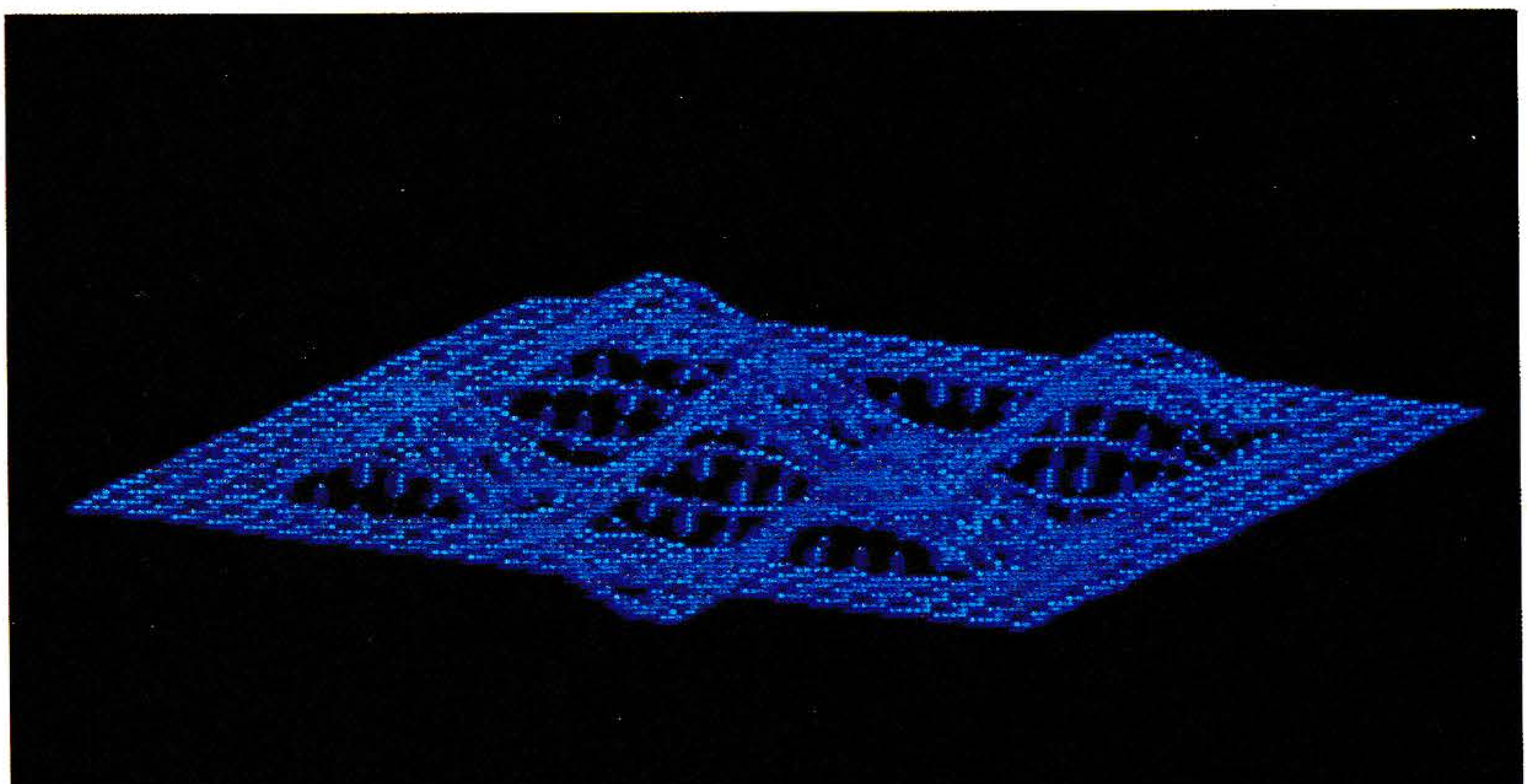


Figure 1



して左上方へ並べていくという操作です．これによって画面に奥行きを与えることができるわけですが，この計算を行っているのが，280，290行です．

ここで使っている式がどのようなものであるのかは，まず *Figure 2* のような辺の長さが1の立方体を考えてみてください．この立方体の頂点 P（1，1，1）を画面に表示するには，原点からどれだけずらせばよいのでしょうか．

それには，いったん頭を3次元から2次元に切り換えて，*Figure 3* を見てください．

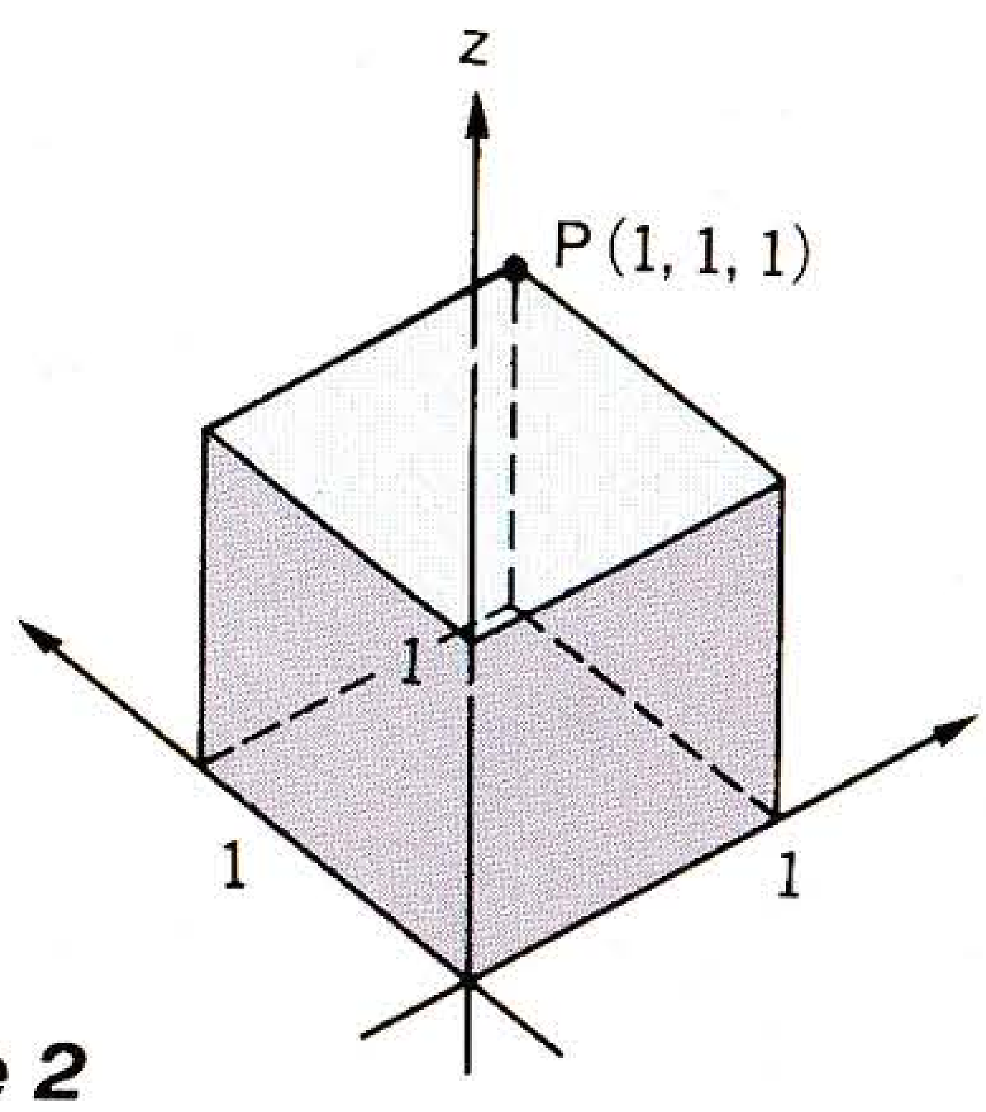


Figure 2

つまり立方体を平面上に描き，そして図中に示した K1～K5の長さを測ります．点 P の平面上での位置は，原点の（GX0，GY0)から右へ K1，上へ K2，左へ K3，上へ K4+K5，だけ離れているわけですから，(GX0+K1−K3，GY0−K2−K4−K5）と表すことができます．

そしてまた，この K1～K5という値によって，もとの立方体で（x，y，z）に位置していた点を画面に表示する場合の座標も，*Figure 4* に示すように計算できます．プログラムの280，290行で使っている計算式は，このようにして導かれたものです．

さて，プログラムを実行させるとわかるように，立体の各点は網目でつながっています．この網目は *Figure 5* に示すように，1つ前のカーブ上のそれぞれの点の位置を，配列 X(I)，Y(I) に記憶しておき，次の位置の点を結んでいくときに，この配列変数の値を参照して，つないで作られているのです．このように3次元の物体を網目によって表す方法は，ワイヤ・フレーム法と

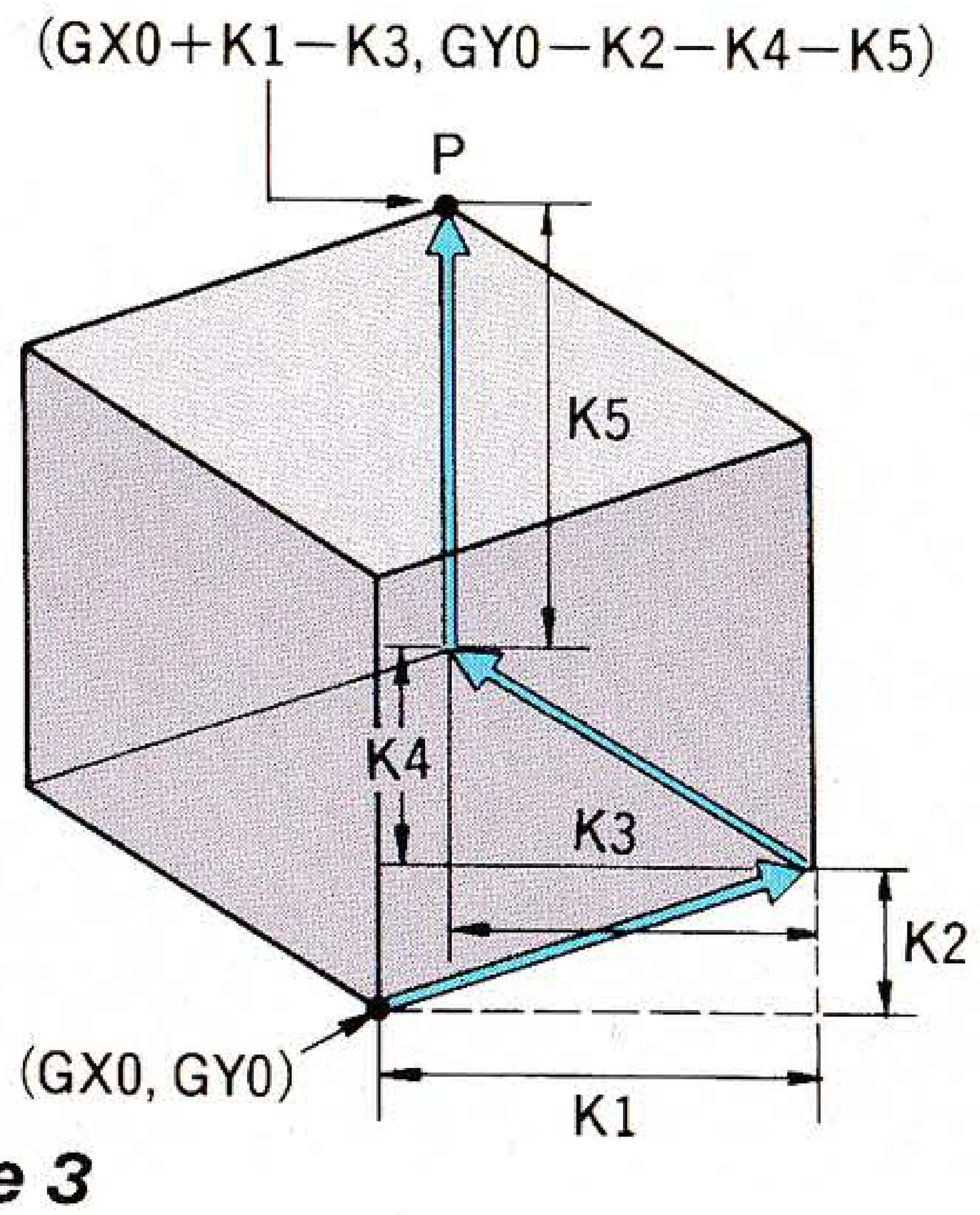


Figure 3

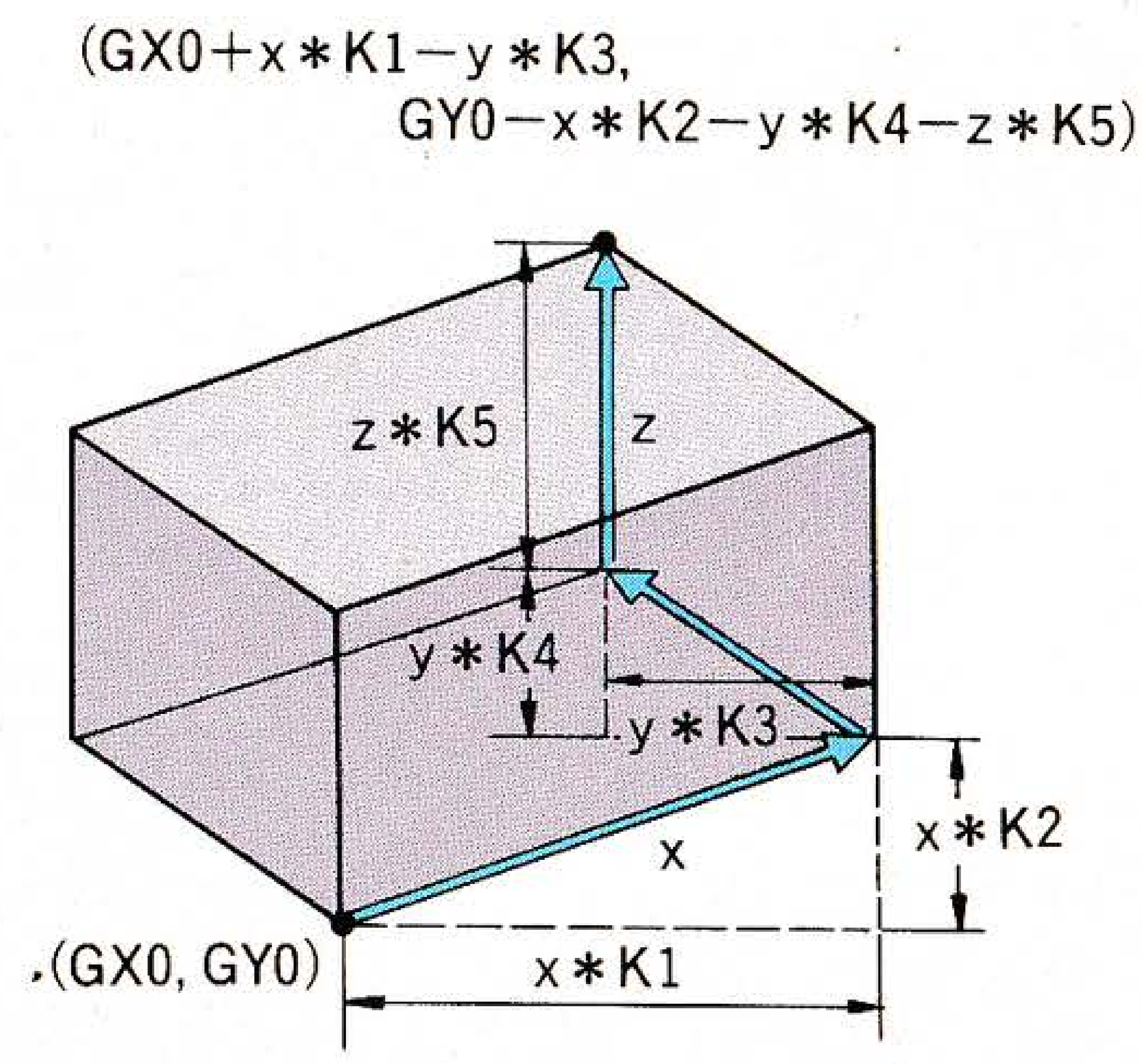


Figure 4

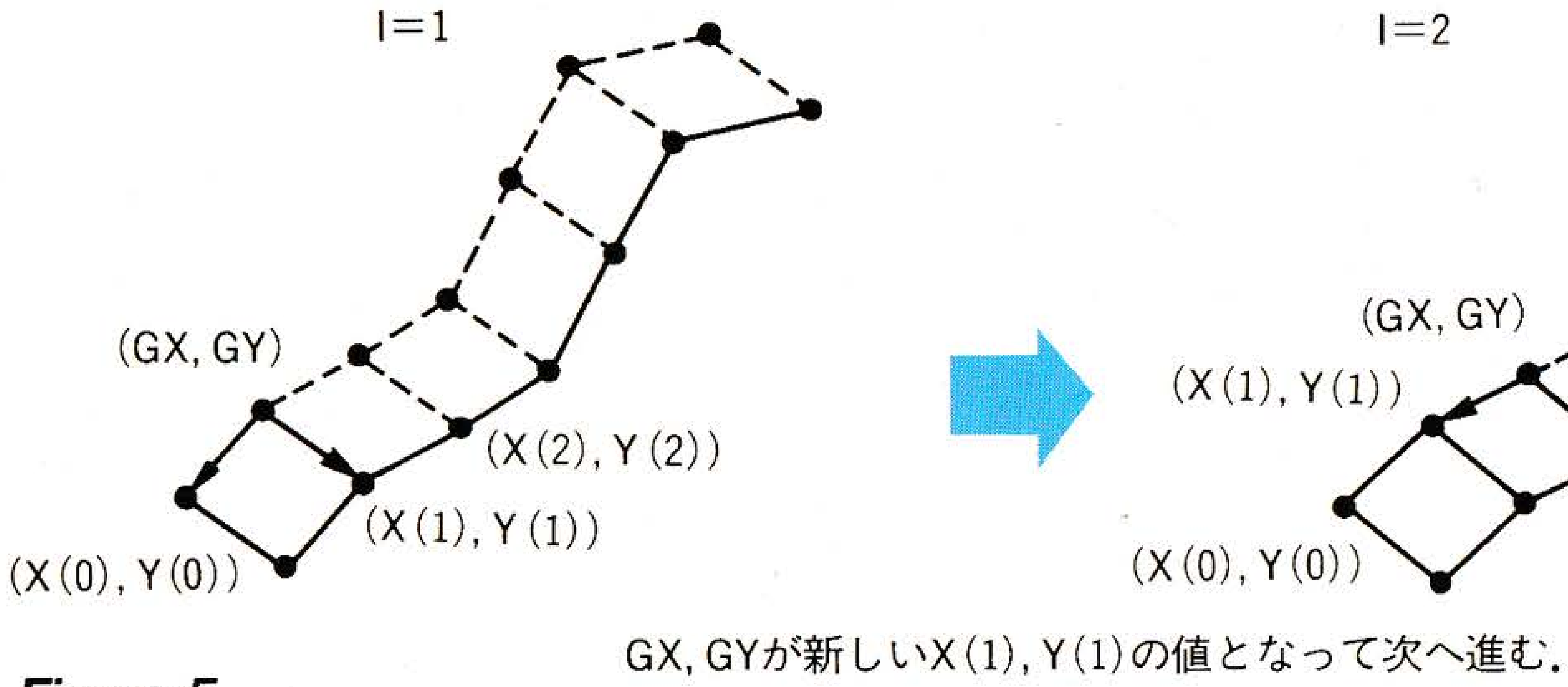


Figure 5

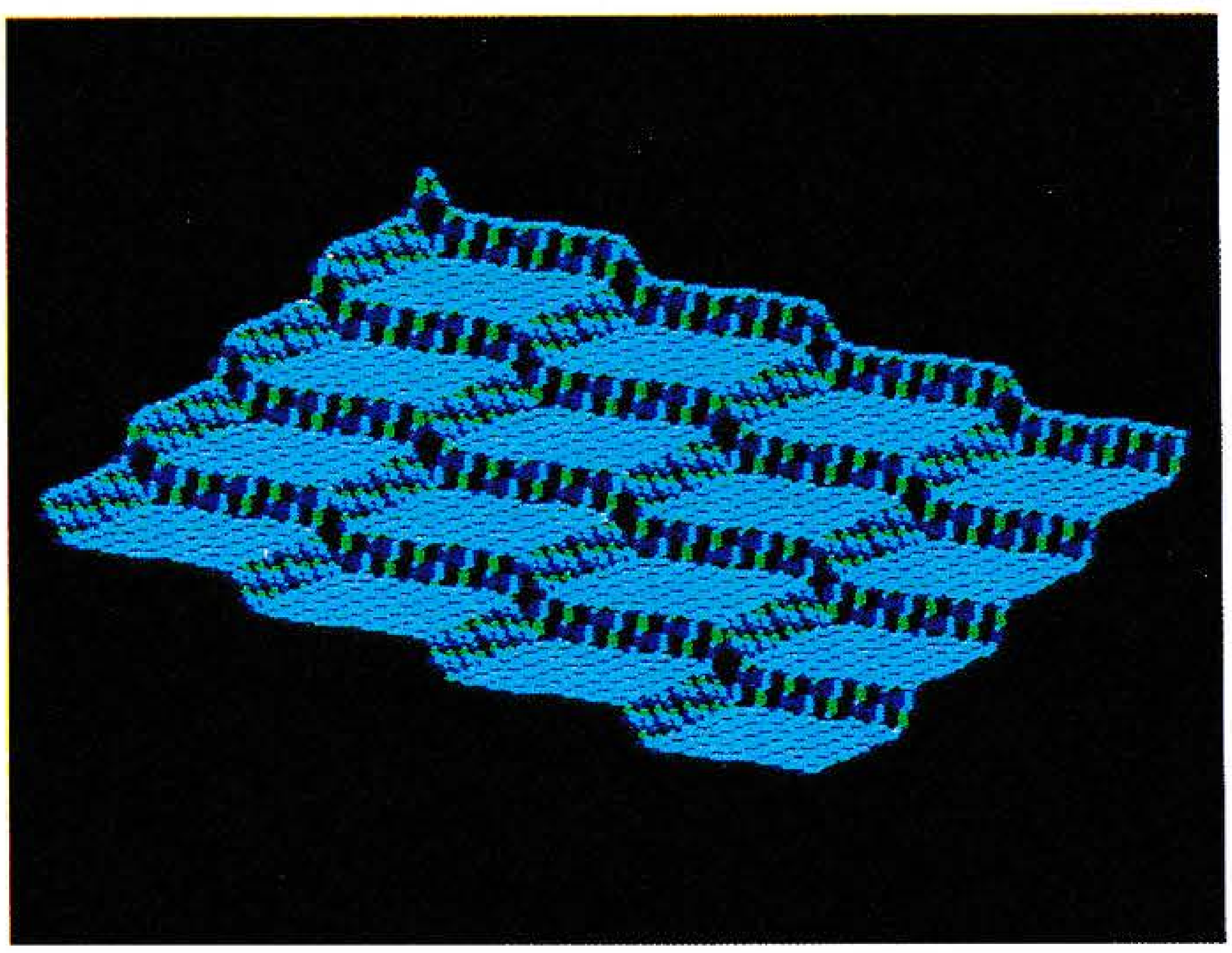
呼ばれます．

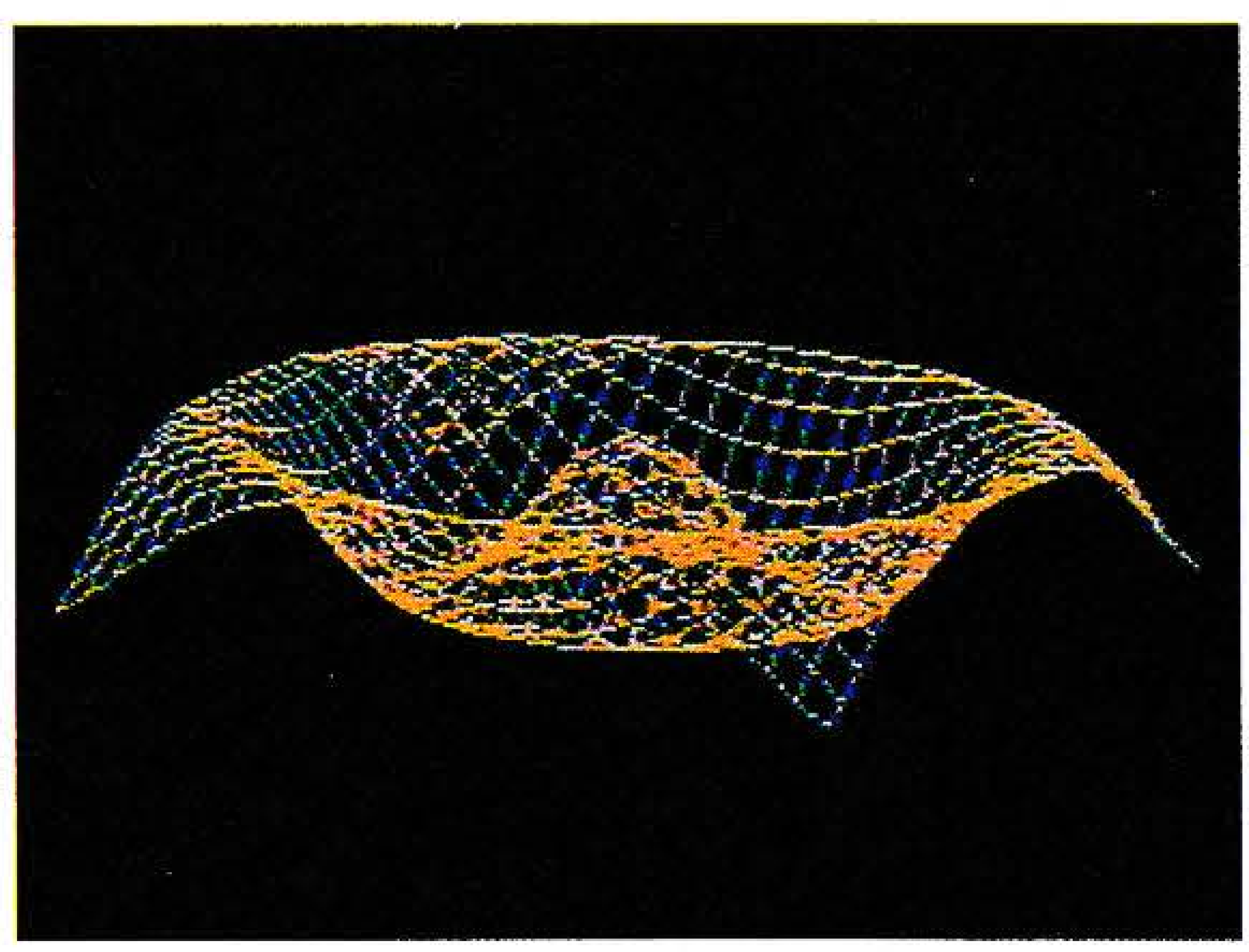
ここで，90ページの写真を見てください．どこか奥行きがないように感じられませんか？ふつう手前にあるものほど大きく見えなければならないのですが，このプログラムでは，手前に見える山（こぶ）も奥に見える山（こぶ）も同じ大きさで表しているため奥行きがそれほど感じられないのです．この奥行きも考えに入れた3Dグラフィックスは，「Appendix　三次元グラフィック・パッケージ」で紹介します．

このプログラムは，設定を変えれば他の立体を表示することもできます．最後にその方法を紹介しておきましょう．

表示される図形は160行の関数定義命令で定義しています．そして表示の範囲を決めているのが140行のARという変数です．x，y両座標とも，0を中心として－ARから＋ARまでが計算され，表示されます．ただし，その範囲内で関数FNZ(X，Y)の値が±2を超えると画面外に線が飛び出してしまってエラーになることがあります．

おもしろい形の立体が表示される設定の例を*List 11a*，*b*に示しました．おためしください．





List 11

```
100 ' List   11
110 '
120 DEFSNG A-Z
130 IE=25 : JE=25
140 AR=6
150 PI=3.14159 : C=8
160 DEF FNZ(X,Y)=COS(SQR(X*X+Y*Y)*PI/3) ………表示するグラフィックを表す関数
170 '
180 DIM X(IE),Y(IE)
190 XS=2*AR/IE : YS=2*AR/JE
200 K1=40/AR : K2=20/AR : K3=80/AR : K4=12/AR : K5=20
210 '
220 SCREEN 2 : COLOR 15,1,1 : CLS
230 FOR J=0 TO JE
240   FOR I=0 TO IE
250     X=(I-IE/2)*XS
260     Y=(J-JE/2)*YS
270     Z=FNZ(X,Y)
280     GX=128+X*K1-Y*K3          } ３次元データを２次元データに変換
290     GY= 96-X*K2-Y*K4-Z*K5     }
300     IF I>0 THEN LINE (GX,GY)-(X(I-1),Y(I-1)),C
310     IF J>0 THEN LINE (GX,GY)-(X(I),Y(I)),C
320     X(I)=GX : Y(I)=GY
330   NEXT I
340 NEXT J
350 DM$=INPUT$(1):END
```

List 11a

```
130 IE=32 : JE=32
140 AR=2
150 PI=3.14159 : C=7
160 DEF FNZ(X,Y)=(INT(X)+INT(Y))/2
```

List 11b

```
130 IE=25 : JE=25
140 AR=12
150 PI=3.14159 : C=4
160 DEF FNZ(X,Y)=COS(X*PI/6)*COS(Y*PI/6)*(ABS(X)<10)*(ABS(Y)<10)
```

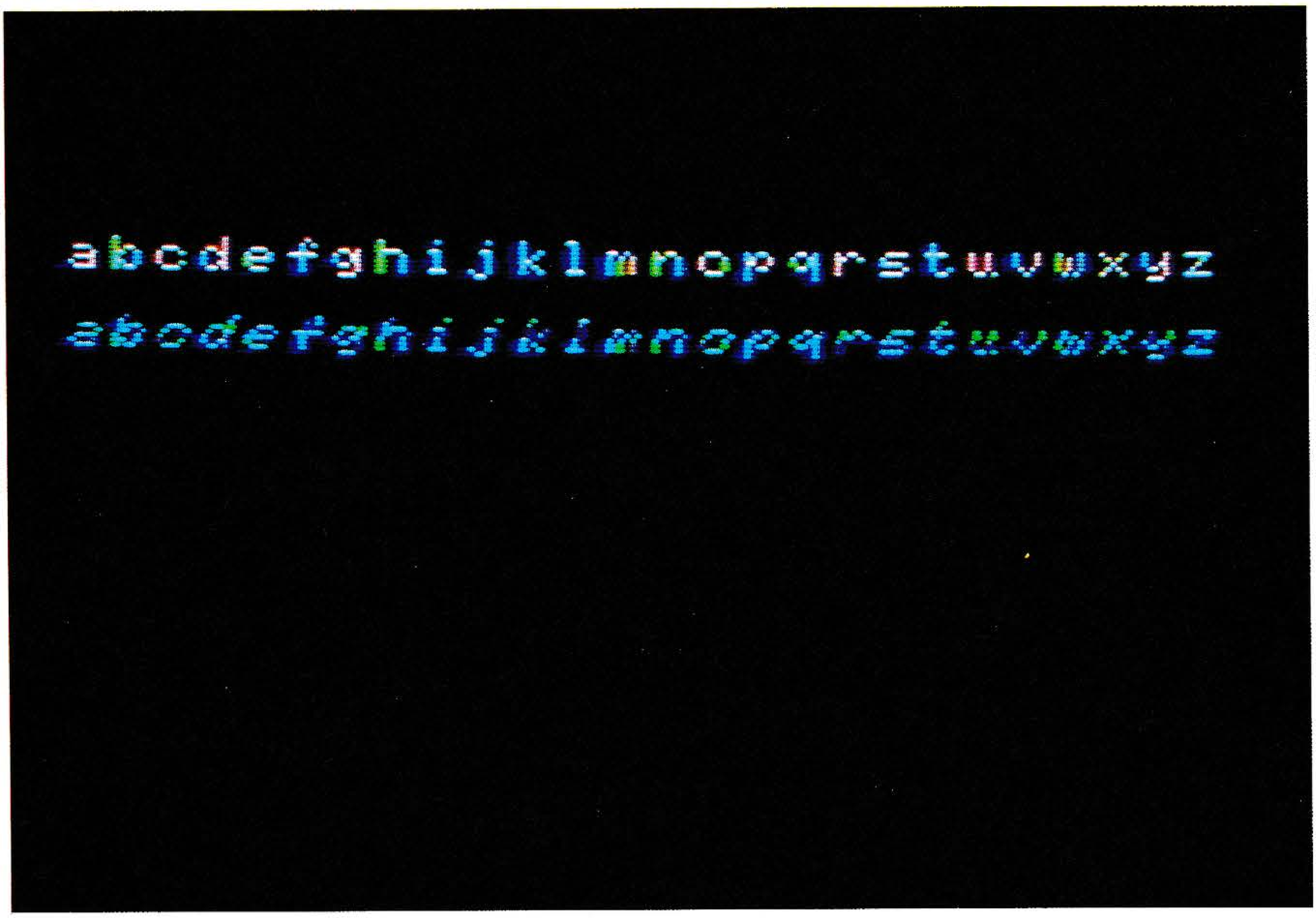
文字の変形

MSX では，文字とグラフィックの両方を1画面上に表示したい場合，グラフィック画面上で文字を扱う必要があります．しかしいったん表示してしまえば，文字とグラフィックをまったく同じ V-RAM 上に出力するという点が，他のマシンとは大きく異なっています．

そのため，一度画面に表示された文字は，そのままグラフィックとして，ドット単位に分けて取り扱うこともできます．

たとえばここに示したプログラムでは，PRINT #命令によって出力し，この文字をデータとしそれを斜めに傾けてイタリック体の文字を作り出してみました．このようなことができるのも，MSX が文字とグラフィックを画面上でまったく区別していないためです．

グラフィック画面（SCREEN 2および



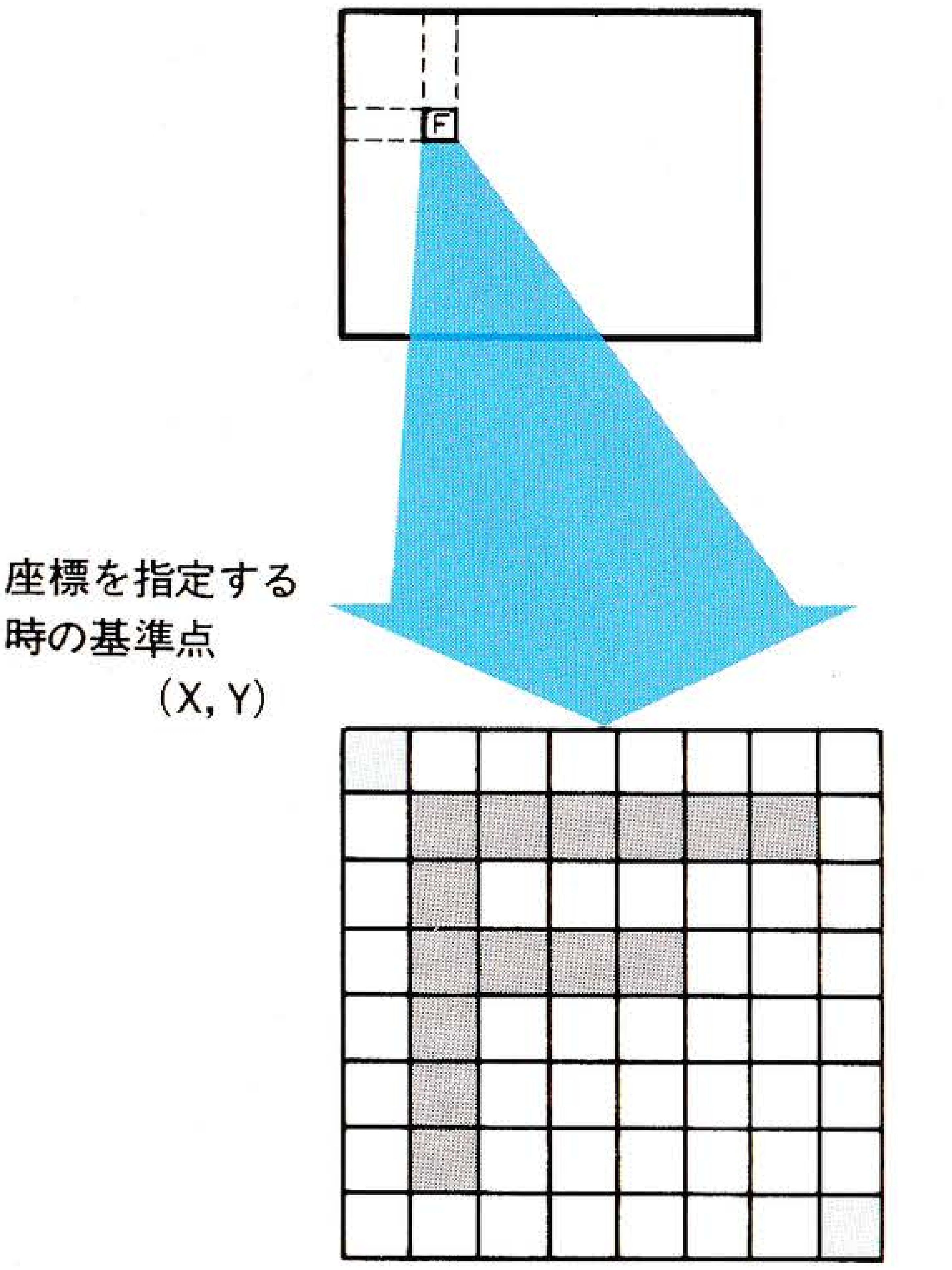


Figure 1

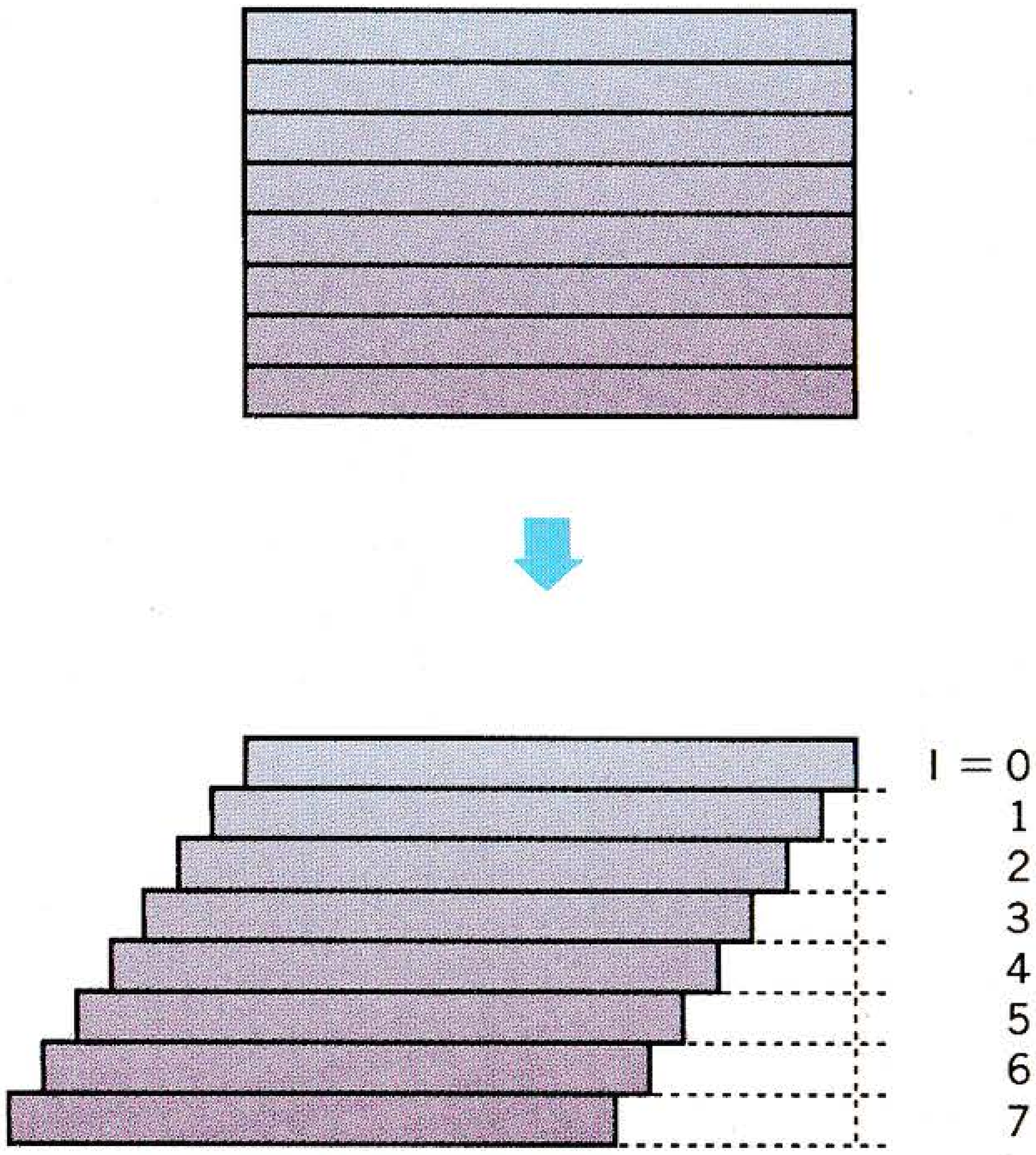


Figure 2

3)に文字を描く時には，コンピュータにグラフィック画面上で文字を扱うことをあらかじめ告げておく必要があります．これが170行のOPEN "GRP:" FOR OUTPUT AS #1という命令です．次に200行のようにPRINT #1によってグラフィック画面上へ文字を表示します．このPRINT #1命令は，表示する画面がグラフィック画面であることを除けば，PRINT命令と同じと考えてよいでしょう．

ここで，文字を画面の真ん中へ表示するときのことを考えてください．テキスト画面では，

```
LOCATE 16,12:PRINT"X"
```

のように1文字単位で位置を指定すればよいのですが，グラフィック画面に文字を表示する際は，ドット単位に指定することになります．

Figure 1 を見てください．文字1つは，8×8ドットで構成されています．この文字をグラフィック画面に出力する場合には，その左上すみの座標を基準点とし，DRAW命令などを利用して，

```
DRAW"BM=x座標;,y座標":
PRINT"X"
x座標16×8,y座標12×8
```

のように位置指定します．この処理は，190行で行っています．

さて，表示された文字は，*Figure 1* に示したようにグラフィック座標の(X，Y)，(X+7，Y+7)という2点を対角線とする四角形の中に描かれています．したがって，この四角形の中を調べれば，文字のドット構成であろうとなんであろうと知ることができるのです．

プログラムの210〜270行ではこのような四角形領域のデータをPOINT命令により読み出し，それを16ドット下にPSET命令でコピーするという作業を行っています．ただし，読み出すときにデータを横にスライスして8つに分割し，それを出力するときには少しずつ横にずらして積み重ねていきます（*Figure 2* 参照）．すると，結果としてもとの四角形が斜めに傾いて表示されるわけです．

傾きの加減を決めているのが220行の代入文です．Nという変数は上から何番目のスライスであるか示すものですが，このNが増えるに従って，N＊0.3だけ表示位置を左にずらしています．この0.3を，もっと大きな値に変えれば傾き具合も大きくなりますし，小さくすれば傾き具合も小さくなります．

List 12

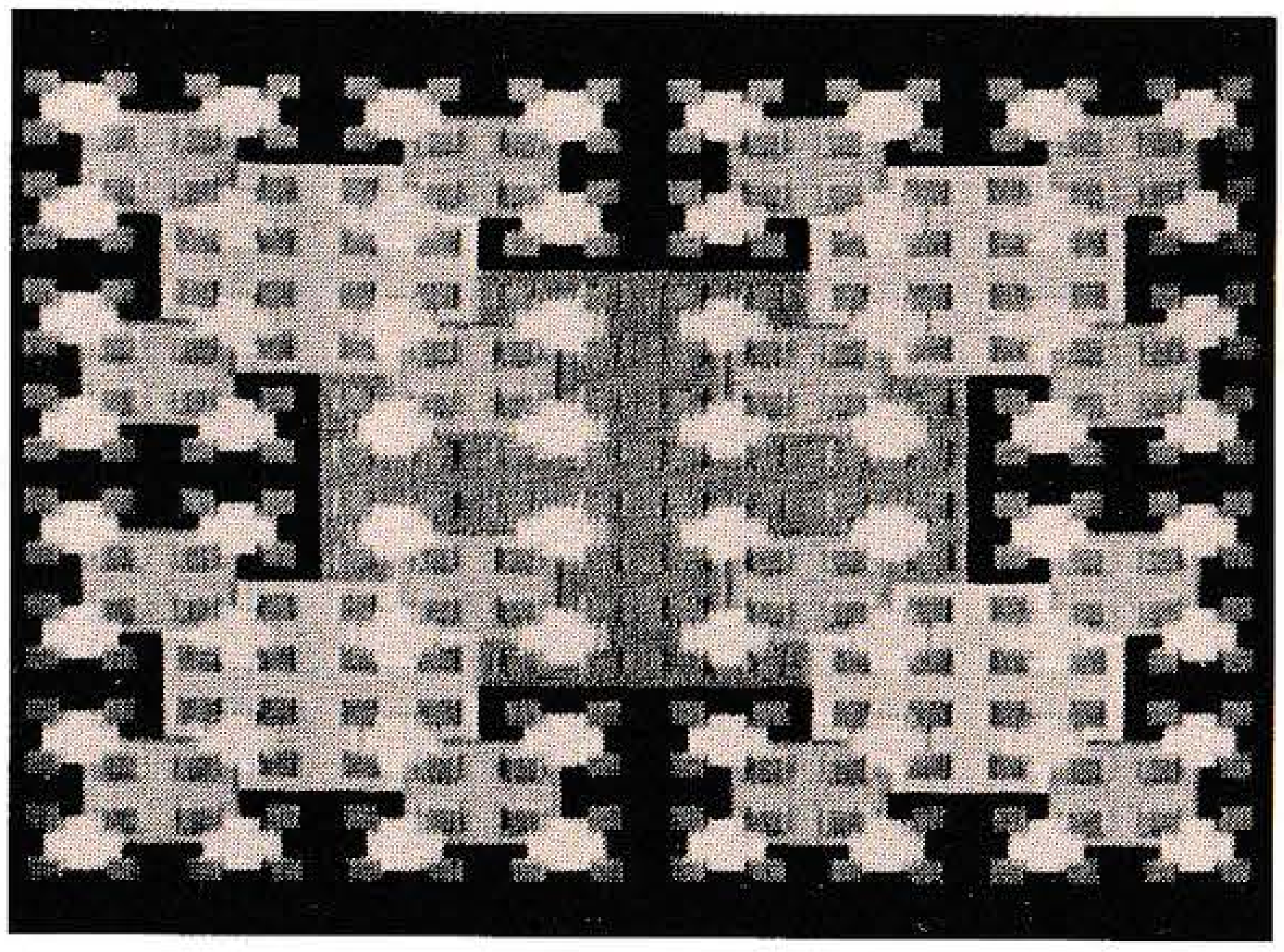
```
100 ' List   12
110 '
120 DEFINT A-Z
130 X=25 : Y=64 : YN=80
140 C=7
150 '
160 SCREEN 2 : COLOR 15,1,1 : CLS
170 OPEN "GRP:" FOR OUTPUT AS #1
180 FOR I=ASC("a") TO ASC("z")
190   DRAW "BM=X;,=Y;"
200   PRINT #1,CHR$(I) ………グラフィック画面に文字を表示する
210   FOR N=0 TO 7
220     S=(7-N)*.3
230     FOR M=0 TO 7
240       P=POINT(X+M,Y+N)
250       IF P<>1 THEN PSET (X+M+S,YN+N),C
260     NEXT M
270   NEXT N
280   X=X+8
290 NEXT I
300 CLOSE #1:DM$=INPUT$(1):END
```

（230〜260行）表示されている文字を変形して表示し直す

リカーシブ(再帰)

ここに挙げたのは，再帰的な手法を使ったグラフィックのプログラムです.

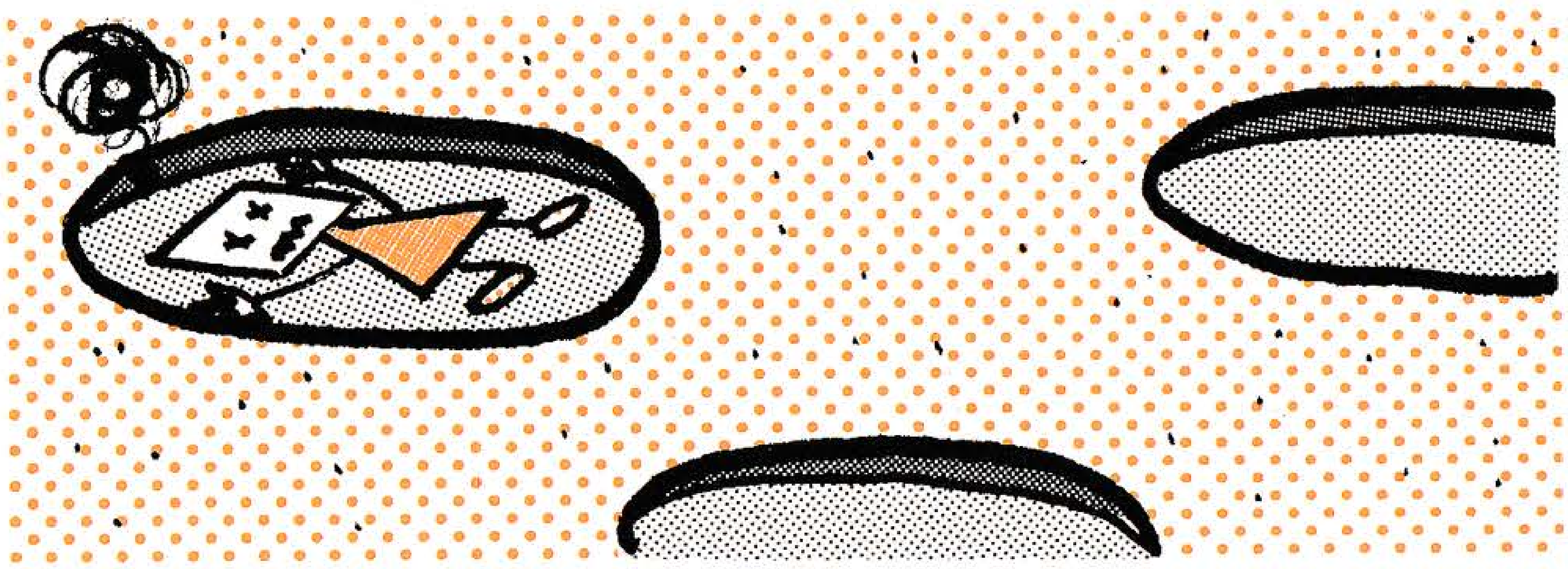
実行画面を見ているとわかりますが，このプログラムはまず四角形を描き，その4つの頂点を中心にそれぞれ小さな四角形を描き，それらの頂点を中心にさらに小さな四角形を描き……という操作を5回繰り返しています.



```
100 ' List Ex.3
110 '
120 SCREEN 2 : COLOR 3,1,1 : CLS
130 DIM XS(5),YS(5)
140 X=128 : Y=96 : XW=128 : YW=96 : C=1
150 '
160 M=5
170 C=(C+1) MOD 10 : L=0 : GOSUB 220
180 GOTO 160
190 '
200 ' ------- recursive
210 '
220 IF L=M THEN RETURN
230 LINE (X-XW/2,Y-YW/2)-(X+XW/2-1,Y+YW/2-1),L+C,BF
240 L=L+1 : XS(L)=X : YS(L)=Y
250   XW=XW/2 : YW=YW/2
260   X=XS(L)-XW : Y=YS(L)-YW : GOSUB 220
270   X=XS(L)+XW : Y=YS(L)-YW : GOSUB 220
280   X=XS(L)-XW : Y=YS(L)+YW : GOSUB 220
290   X=XS(L)+XW : Y=YS(L)+YW : GOSUB 220
300   XW=XW*2 : YW=YW*2
310 X=XS(L) : Y=YS(L) : L=L-1
320 RETURN
```

【4章】

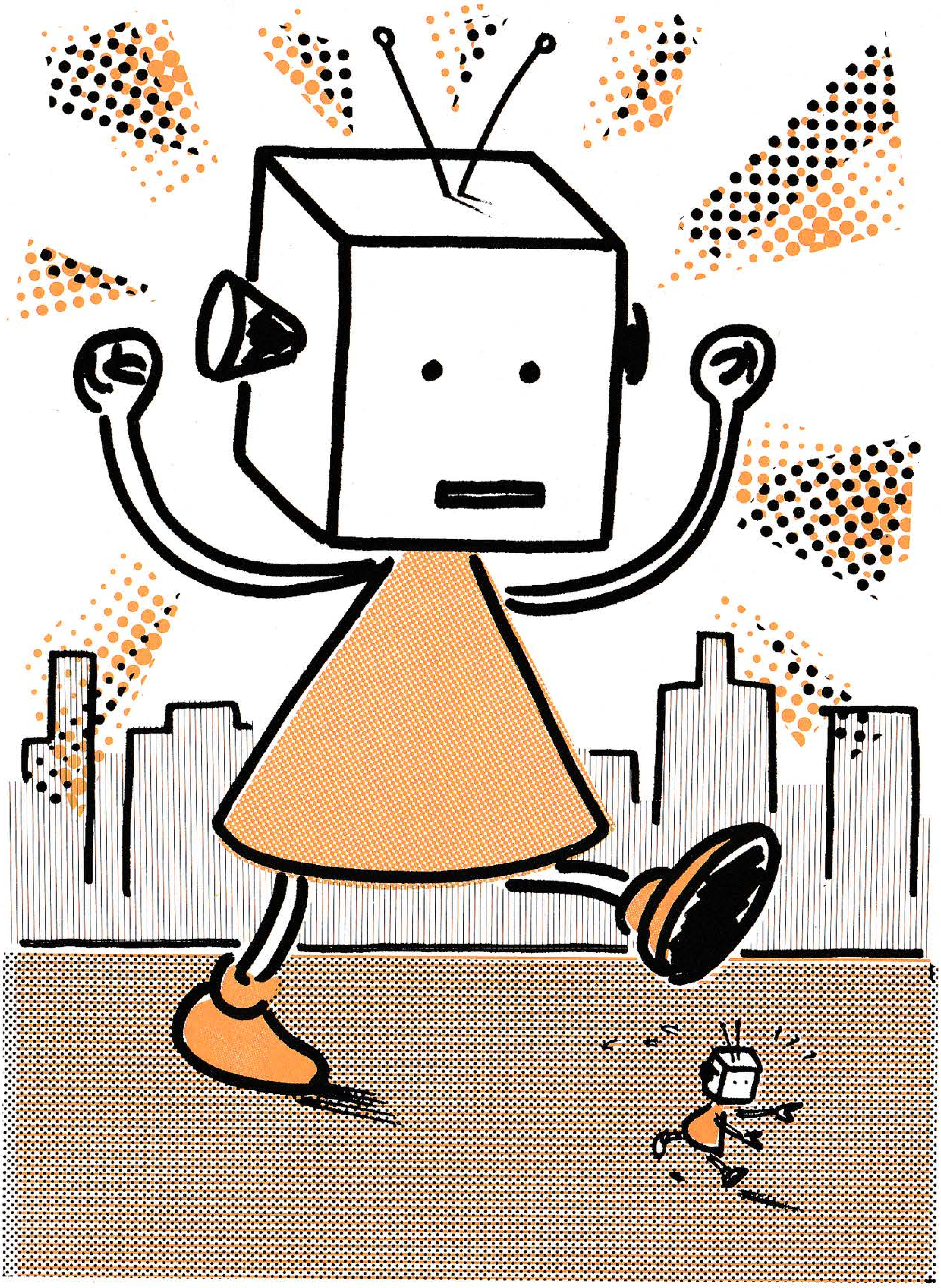
ゲームへのアプローチ



テニスゲームの仕組み
ハイスコアの表示
ビーム砲を打つ
基地が危ない!
図形の拡大,縮小
ストップウォッチを作る
迷路を作る
音符とともに音楽を!
飛行機を飛ばす
タイマー&エネルギーメータ
流れる星空
カニさんカニさん

本章では,BASICプログラムによるTVゲームの可能性を追求してみたいと思います.

前章では画面に描かれたアートを見て「美しい……」と感心するのが主たる目的でした.しかし単なる静止画面のきれいさだけではTVゲームはできません.加えて動きの面白さが必要になってくるのです.グラフィックに動きを与えること,これが本章の目的です.



テニスゲームの仕組み

ゲームの中には，飛行機を上下に動かして敵の砲撃をかわす，ラケットを左右に動かしてボールを打ち返すなど，キーボードからの入力で物を動かす場面が数えきれないほどあります．

そこで，この「キーボードからの入力で物を動かす」例をお見せしましょう．まずはシンプルに，パドルの移動を扱ってみました．

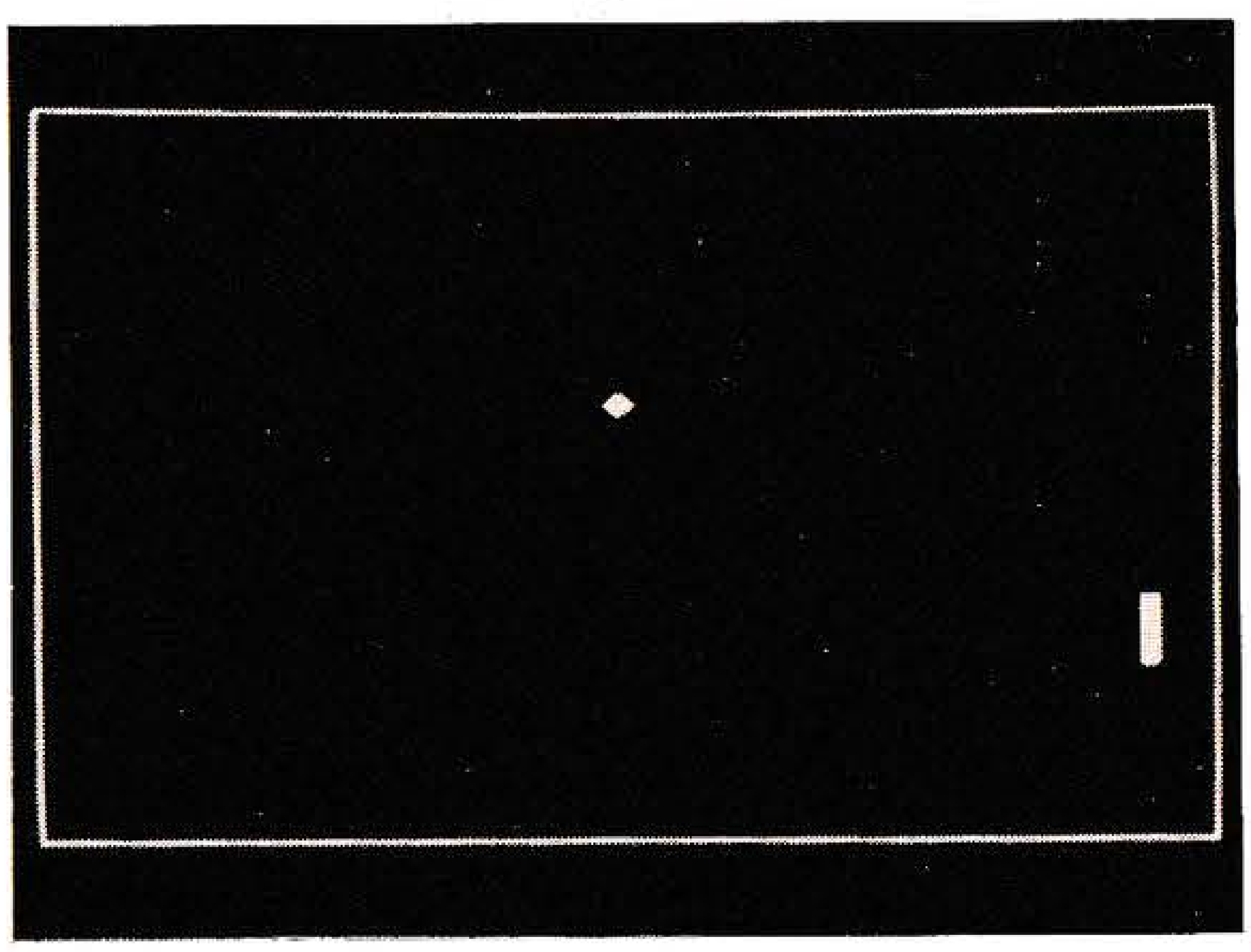
パドルを動かすには，パドルの形をスプライトパターンに設定したのち，

(1) キーボードからの入力待ち
(2) どんなキーが押されたのか判定
(3) パドルをスプライト面へ表示

といったルーチンを繰り返し，スプライト機能を利用して，パドルを動かします．

List 1a を見てください．180行のSCREEN命令により16×16ドットとしています．このスプライトパターンを，160，190行で定義したのち，210行のSTICK(0)でキーボードからの入力を受けつけます．もしカーソルキー⇧が押されたら，220行で判定しパドルを4ドット分上へ置き直します．カーソルキー⇩が押されたら，230行で判定し，パドルは4ドット分下へ置き直します．

カーソルキー⇧，⇩のどちらかが押されると，270行以下のサブルーチンへ分岐します．分岐してパドルを置き直すときに，指定画面外にパドルを置いては困ります．そこで，270，280行に，パドルを置く位置(Y1+V)が画面外になるようなら置き直しを行わないという条件文が必要になるのです．また，実際にパドルを置き直しているのは290行のPUT SPRITE命令です．なお，このプログラムは⇦または⇨キーにより終了します．



さてこれでパドルを動かすことはできました．次はボールを動かしてみましょう．ボールを動かすのも，原則的にはパドルを動かすのと同じですが，ボールが壁に当たったときの処理については新たに考えなければなりません．そこで，ボールが壁に当たって反射する仕組みについて考えてみます．

たとえば *Figure 1* のようにボールは，右上へ移動して上側の壁に当たったときに反射する方向が右下へと変わります．また

ボールは，右下へ移動して右側の壁に当たったときに反射する方向が左下へと変わります．要するに，ボールが当たったと判断したときに，ボールの進行方向を変えてやればよいのです．

ボールが上下いずれかの壁に当たったときにはBY，左右いずれかの壁に当たったときにはBXの正負を逆にしてやればよいというわけです．

たとえば *Figure 2* のように，ボール

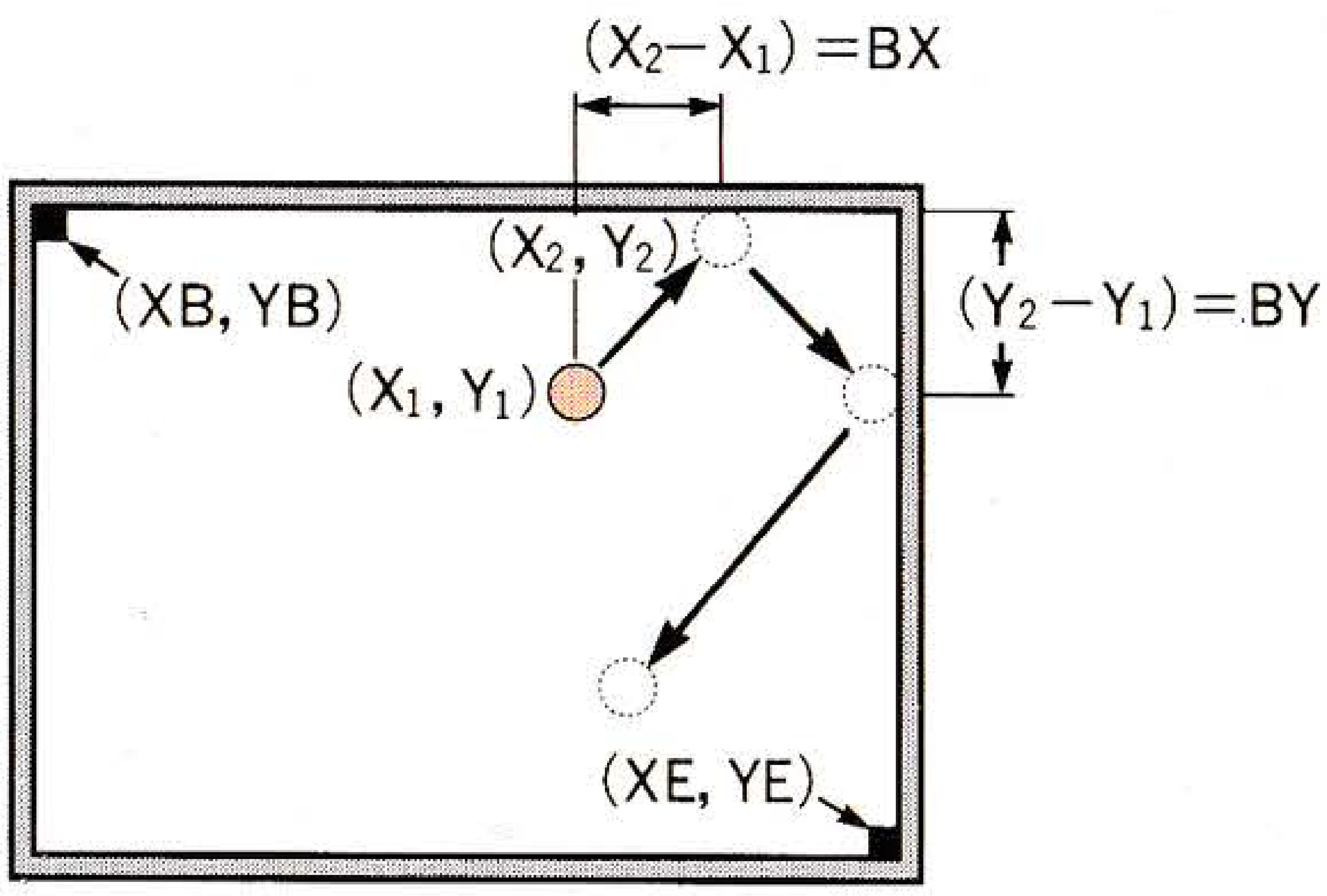


Figure 1

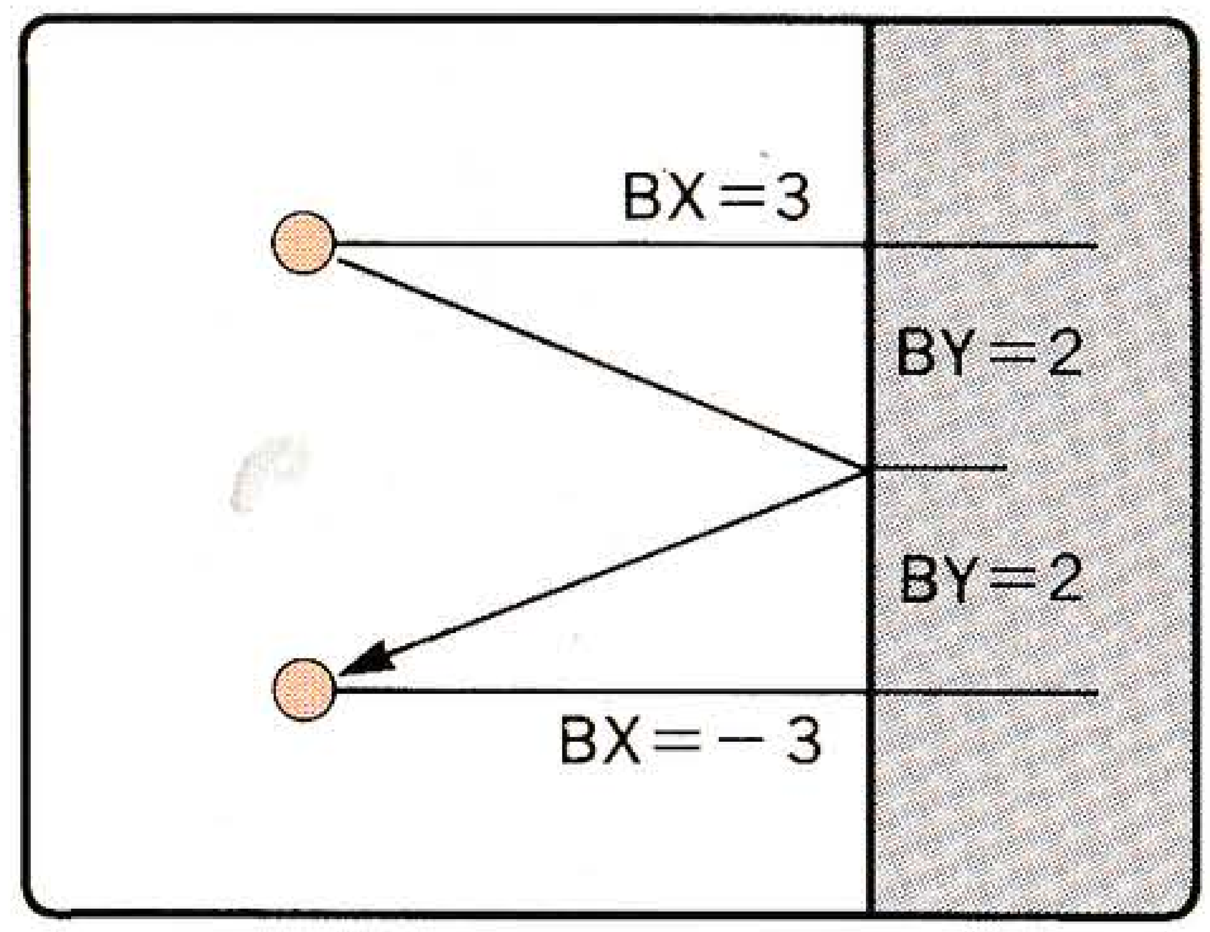


Figure 2

List 1a

```
100 ' List   1a
110 '
120 DEFINT A-Z
130 XB=10  : YB=10
140 XE=250 : YE=180
150 BX=3   : BY=2
160 A$=STRING$(8*2,CHR$(240))+STRING$(8*2,CHR$(0))……スプライトパターン・データの設定
170 '
180 SCREEN 2,2 : COLOR 15,1,1 : CLS
190 SPRITE$(0)=A$
200 X1=XE-16 : Y1=(XB+XE)/2 : GOSUB 270
210 I=STICK(0)
220   IF I=1 THEN V=-PY : GOSUB 270
230   IF I=5 THEN V=PY  : GOSUB 270     押されたキーの判定
240   IF I=3 OR I=7 THEN END
250 GOTO 210
260 '
270 IF V=-PY AND Y1+V<YB THEN RETURN
280 IF V=PY  AND Y1+V>YE-16 THEN RETURN     パドルの移動
290 Y1=Y1+V : PUT SPRITE 0,(X1,Y1),15,0
300 RETURN
```

[←] [→] このキーを押すとプログラムは終了します．

が BX=3，BY=2で移動して右の壁に当たったときには BX=−3，BY=2となり，縦方向の進み方は変わりませんが，横方向の進み方は反対になります．

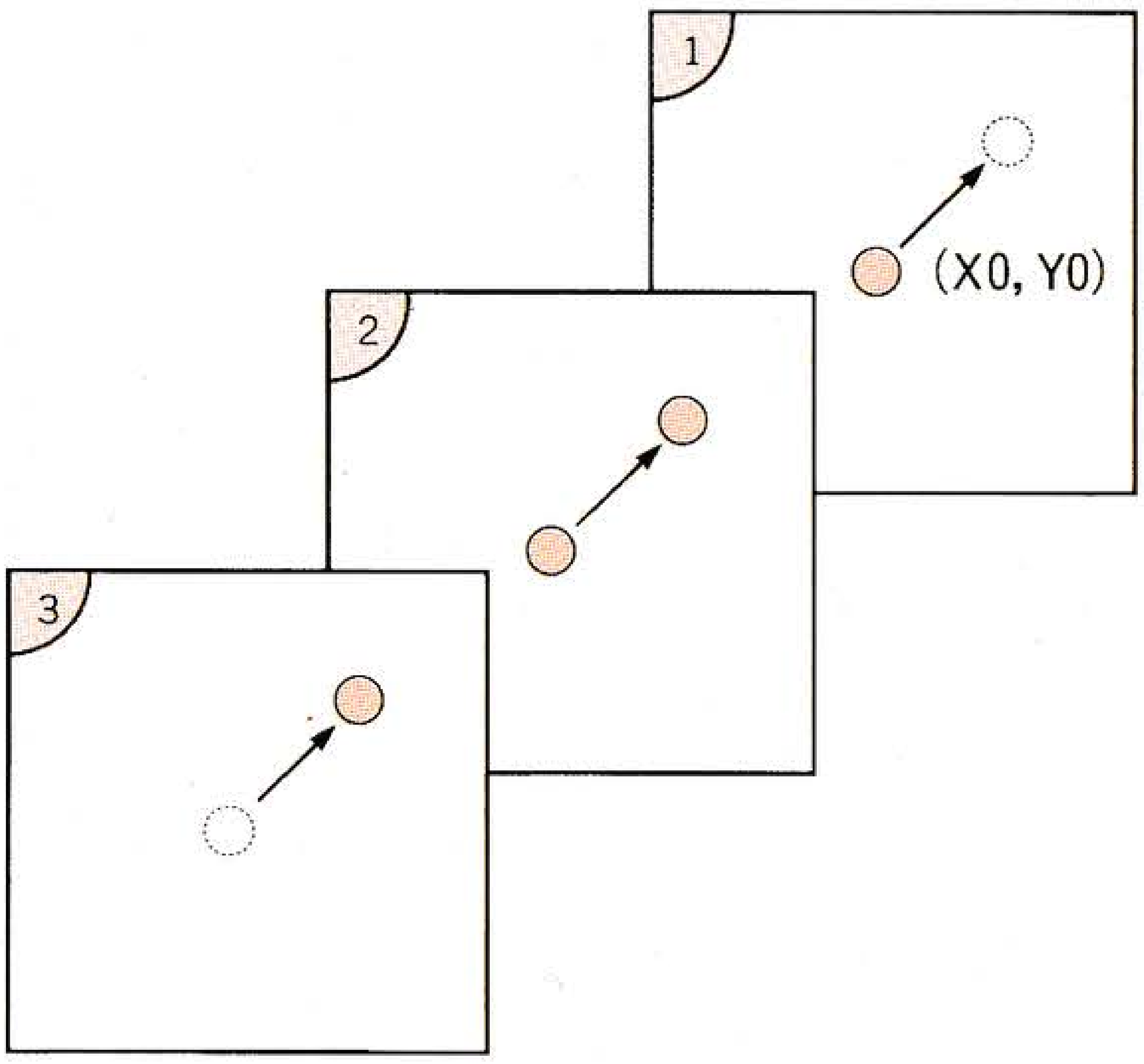


Figure 3

ここで *List 1b* を見てください．壁に当たったときの処理は，240，250行で行っており，ボールを置き直しているのは，270行です．パドルと同様に，スプライトを利用してボールを動かしているのです．

なおこのプログラムの終了は，[CTRL]＋[STOP] キーを押してください．

List 1a，*List 1b* を応用して，ボールをパドルで受けられるようにしてみました．*List 1c* を見てください．260，280，450〜480行が新たに追加された部分です．画面に表示されたパドルとボールのスプライトパターンが重なると，ON SPRITE GOSUB 命令によって450行以降のサブル

List 1b

```
100 ' List   1b
110 '
120 DEFINT A-Z
130 XB=10   : YB=10
140 XE=250  : YE=180
150 BX=3    : BY=2
160 X0=(XB+XE)/2 : Y0=(YB+YE)/2
170 B$=CHR$(48)+CHR$(120)+CHR$(252)+CHR$(120)+CHR$(48)+STRING$(
26,CHR$(0))  ………スプライトパターン・データの設定
180 PLAY "V12T255"
190 '
200 SCREEN 2,2 : COLOR 7,1,1 : CLS
210 LINE (XB,YB)-(XE,YE),7,B  ………枠を描く
220 SPRITE$(1)=B$
230 '
240 IF X0+BX>XE-5 OR X0+BX<XB THEN BX=-BX   : PLAY "L6406A"
250 IF Y0+BY>YE-6 OR Y0+BY<YB-1 THEN BY=-BY : PLAY "L6406A"
260 X0=X0+BX : Y0=Y0+BY
270 PUT SPRITE 1,(X0,Y0),11,1
280 GOTO 240
```

（240〜280行：ボールの移動）

ーチンを実行し，ボールを反射させます．
なお，BX，BY を変えると，ボールの速さや方向を変えられますから，ためしてみてください．

List 1c

```
100 ' List   1c
110 '
120 DEFINT A-Z
130 XB=10 : YB=10
140 XE=250: YE=180
150 BX=3  : BY=2
160 X0=(XB+XE)/2 : Y0=(YB+YE)/2
170 X1=XE-16 : Y1=(YB+YE)/2 : PY=4
180 A$=STRING$(16,CHR$(240))+STRING$(16,CHR$(0))
190 B$=CHR$(48)+CHR$(120)+CHR$(252)+CHR$(120)+CHR$(48)+STRING$(
26,CHR$(0))
200 PLAY "V12T255"
210 '
220 SCREEN 2,2 : COLOR 7,1,1 : CLS
230 LINE (XB,YB)-(XE,YE),7,B
240 SPRITE$(0)=A$ : SPRITE$(1)=B$
250 GOSUB 400
260 ON SPRITE GOSUB 450 ………パドルとボールが当たったかどうかを判定するために付け加えた部分
270 '
280 SPRITE ON
290 IF X0+BX>XE-5 OR X0+BX<XB THEN BX=-BX   : PLAY "L64O6A"
300 IF Y0+BY>YE-6 OR Y0+BY<YB-1 THEN BY=-BY : PLAY "L64O6A"
310 X0=X0+BX : Y0=Y0+BY
320 PUT SPRITE 1,(X0,Y0),11,1
330 '
340 I=STICK(0)
350   IF I=1 THEN V=-PY : GOSUB 400
360   IF I=5 THEN V=PY  : GOSUB 400
370   IF I=3 OR I=7 THEN END
380 GOTO 280
390 '
400 IF V=-PY AND Y1+V<YB  THEN RETURN
410 IF V=PY  AND Y1+V>YE-16 THEN RETURN
420 Y1=Y1+V : PUT SPRITE 0,(X1,Y1),15,0
430 RETURN
440 '
450 SPRITE OFF
460 PLAY "L64O3A"
470 X0=X0-BX : BX=-BX
480 RETURN
```



ハイスコアの表示

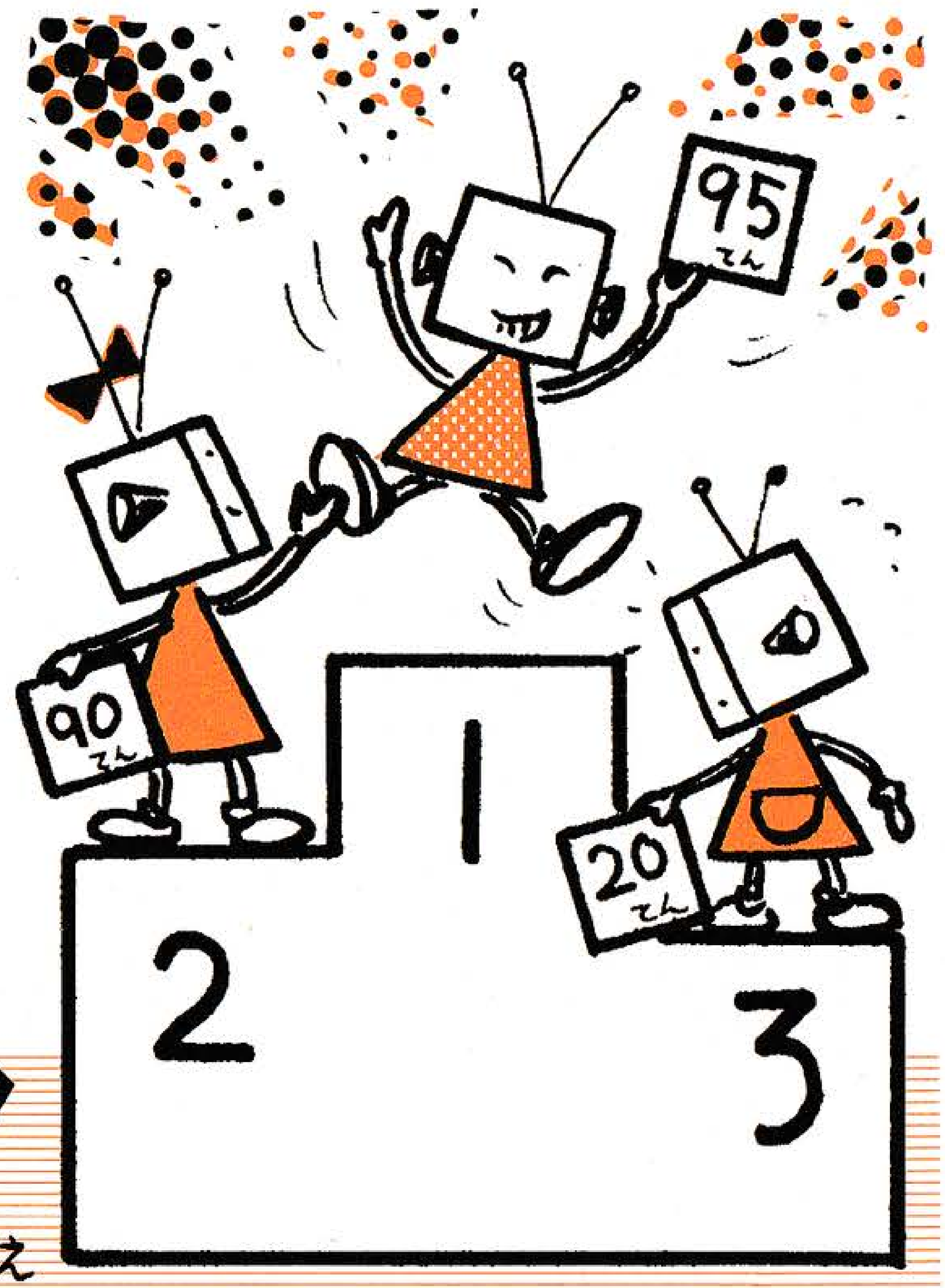
深夜一人でコンピュータ・ゲームを楽しんでいるなんて人，けっこういるんでしょうね．でもそんな人には，このプログラムは大してお役に立ちません．

友達といっしょに，ワイワイ・ガヤガヤとゲームを楽しんでいる人のためのプログラムです．

ゲームが終わった時点で，スコアの高い順に名前を表示したいことがあると思います．こんなときには，スコアが高い順に名前を並べ換えればよいでしょう．

プログラムを見てください．190〜240行が，このスコアの高い順に名前を並べ換えている部分で，*Figure 1* のように2つのデータ（スコア）を比較して逆順ならこの2つのデータを入れ換えるという処理を行っています．この結果から，290〜350行のPRINT，PRINT USING 命令を用いて表を作ります．





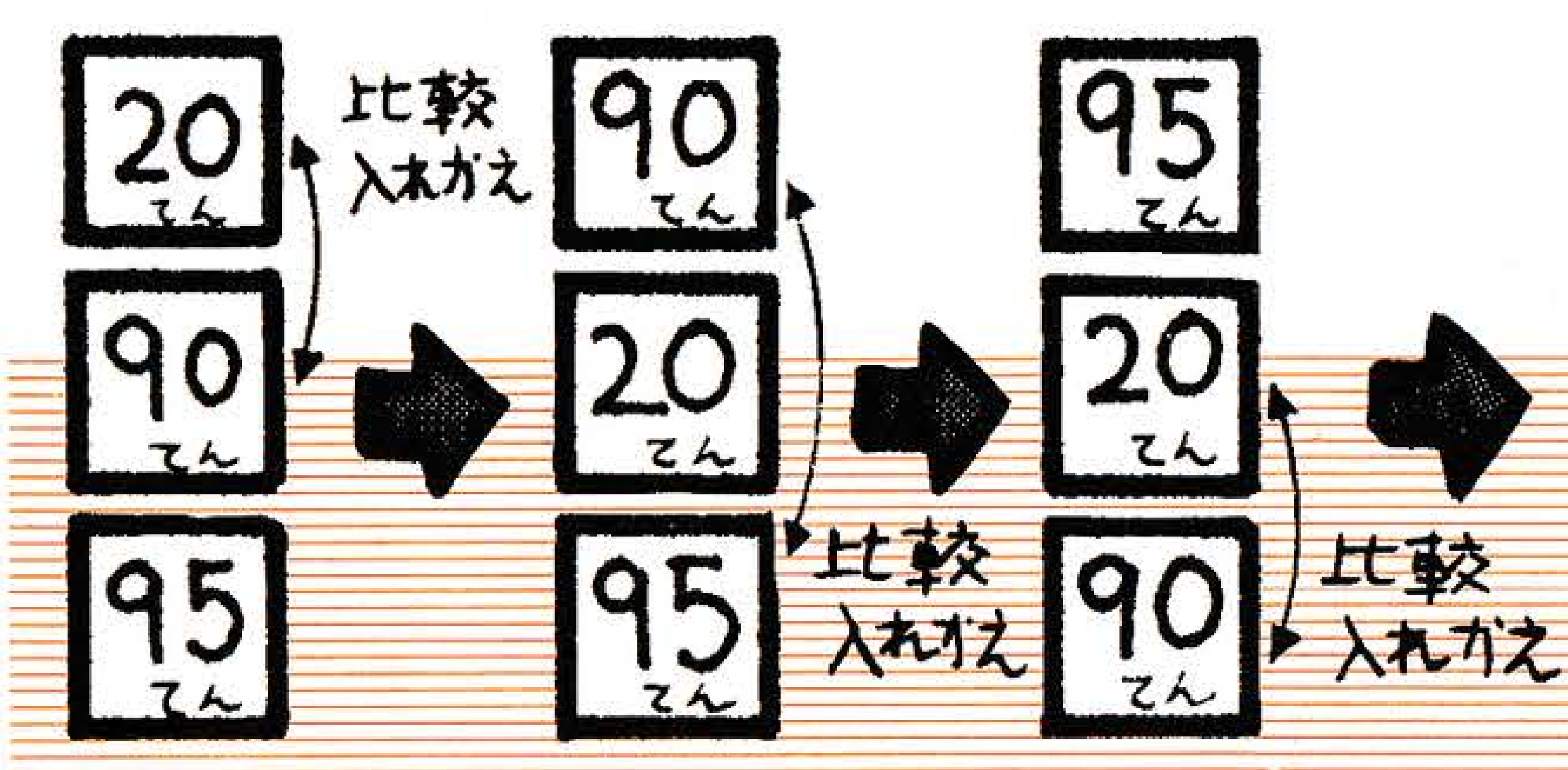


Figure 1

List 2

```
100 ' List   2
110 '
120 DEFSNG A-Z
130 '
140 RESTORE 380
150 READ N : DIM D$(N),D(N)
160 FOR I=1 TO N
170   READ D$(I),D(I)
180 NEXT I
190 FOR I=1 TO N-1
200   FOR J=I TO N
210     IF D(I)<D(J) THEN SWAP D(I),D(J) : SWAP D$(I),D$(J)
220     IF TS<D(I) THEN TS=D(I) : TS$=D$(I)
230   NEXT J
240 NEXT I
250 '
260 SCREEN 0 : COLOR 15,2,2 : CLS
270 KEY OFF
280 '
290 PRINT "No.     Score     Name"
300 PRINT
310 FOR I=1 TO N
320 PRINT USING "##    #######    &       &";I;D(I);D$(I)
330 NEXT I
340 PRINT
350 PRINT USING "Top   #######    &       &";TS;TS$;
360 DM$=INPUT$(1):END
370 '
380 DATA 8
390 DATA "ｱｵﾔﾏ"    ,  847000
400 DATA "ｻｸﾗﾀﾞ"   ,  550000
410 DATA "ﾐﾉｼﾏ"    ,  400000
420 DATA "ｵｼﾞｻﾝ"   ,       0
430 DATA "ﾔｷﾞ"     ,  377700
440 DATA "ﾌｼﾞﾀ"    ,  428398
450 DATA "ｳﾁｺ"     ,   98777
460 DATA "ｵｵｸﾎﾞ"   ,  390999
```

スコアの高い順にデータをならべ換える（190〜240）

表の作製（290〜350）

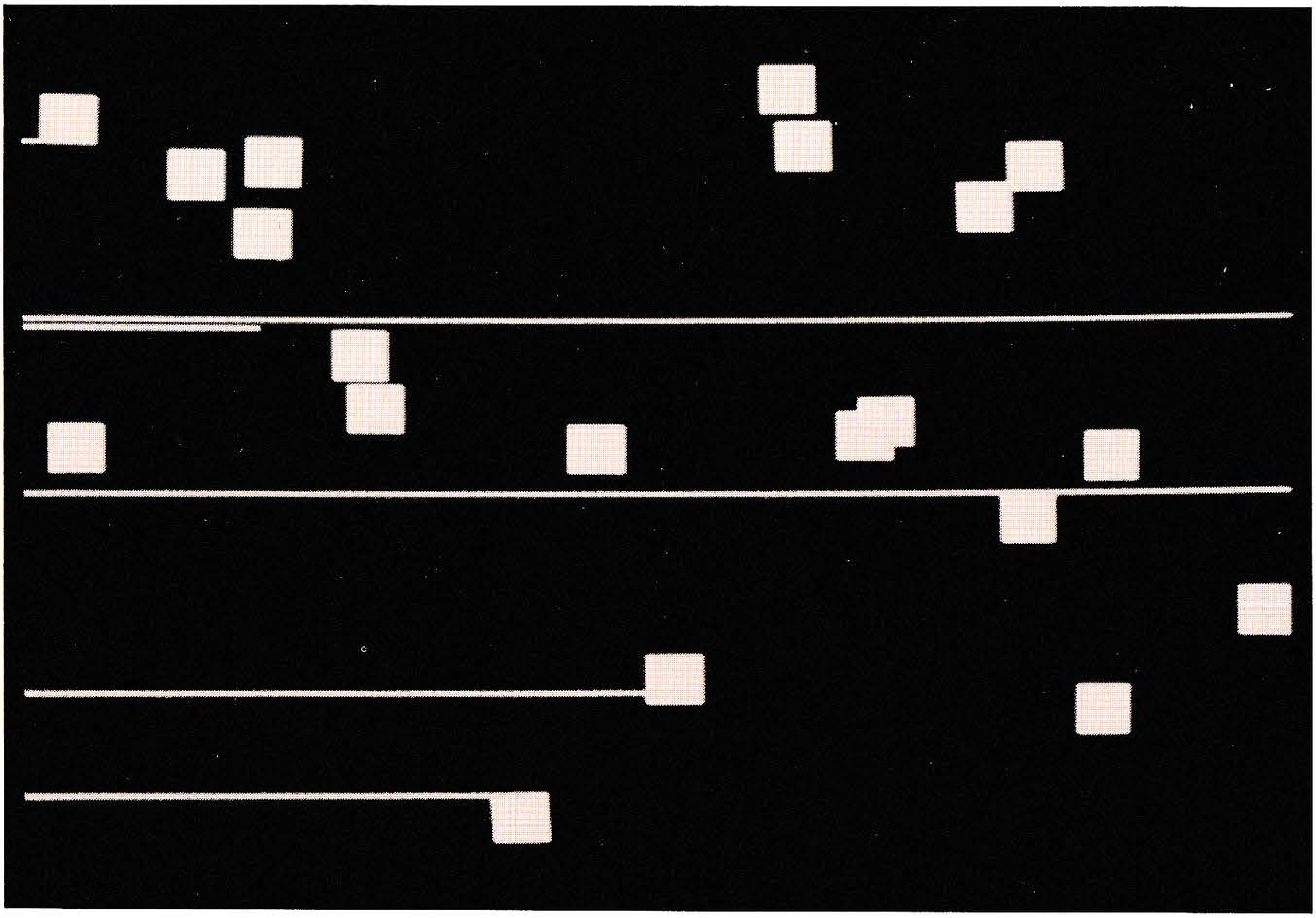
得点データ（380〜460）

ビーム砲を打つ

インベーダーにミサイルが当たったり，敵の基地を破壊したり，ゲームでは「何かが何かに当たる」という状況が，非常に多く見受けられるものです．プログラムでは，どうやって当たりはずれを判断しているのでしょうか．ここで，ビーム砲から発射したビームが物体めがけて飛んで行く場合を考えてみましょう．

物体の数が少なければ，ビーム砲の位置と物体の距離を，そのつど計算してみることもできます．しかし，物体が1万個もあったりすると，ビーム砲との距離を調べているだけで，日が暮れてしまいます．そこで，何かがあるかないかを座標の位置から算出するのではなく，直接画面上のドットを調べて判断するために，POINT 関数が用意されています．

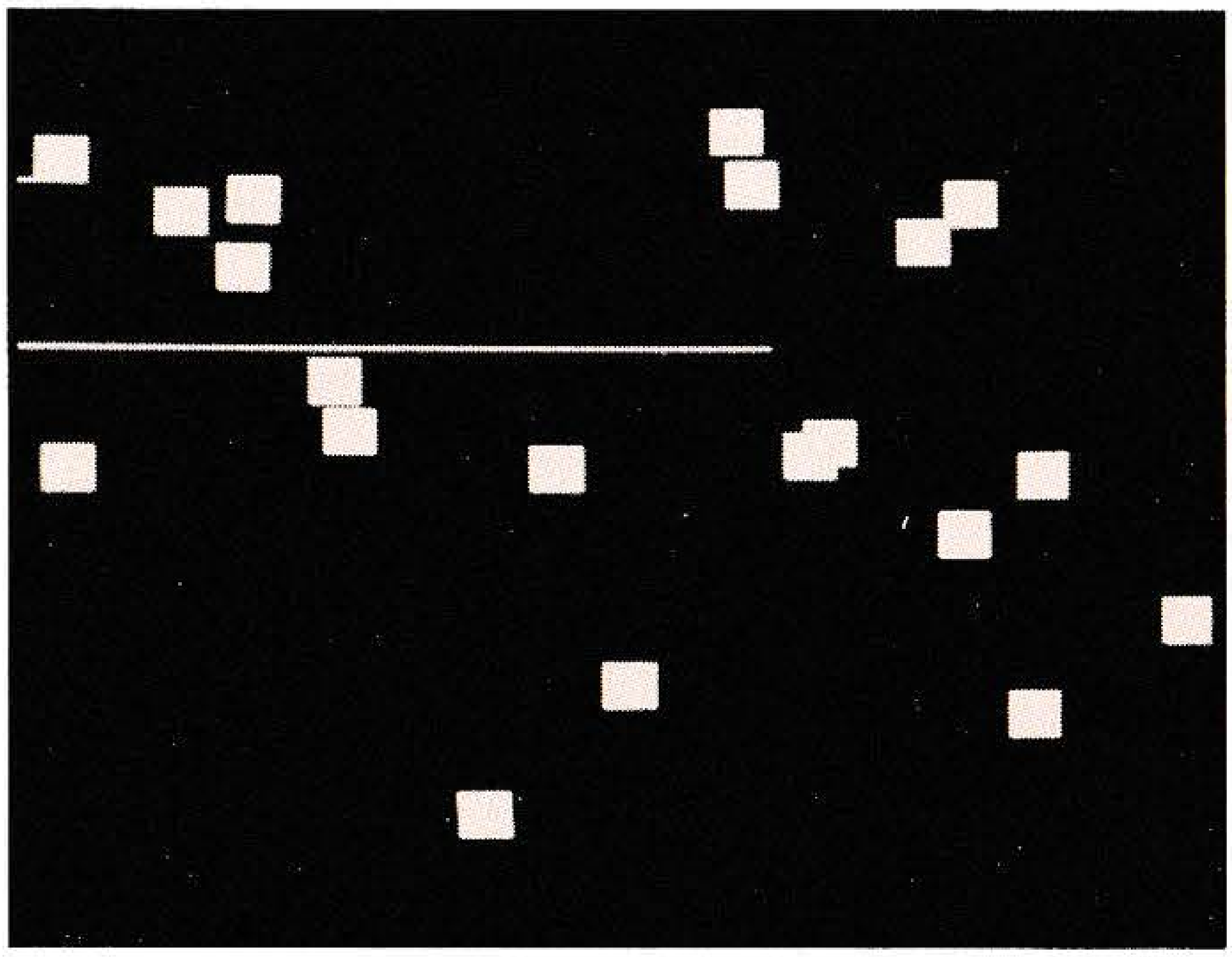
POINT (X，Y) は，もしグラフィック



座標(X，Y)に何か書いてあればカラーコードを，何もなければ0を返してきます．そこで，POINT 関数により1つ先の点を調べて，何もなければビームを1ドット進め，何かあれば命中したとみなすというわけです．

ここでプログラムを見てみましょう．180〜230行で的となる物体をランダムにちりばめています．260〜310行のループで，1ドット先を POINT 関数で調べながら進んでいきます．もしビームが，画面の右端に届いてしまったら，的には当たらなかったことになります．

この方法では，非常にゆっくりとビームが進んで行きます．速くするためには，160行の XS を5としてみてください．



List 3

```
100 ' List   3
110 '
120 DEFINT A-Z
130 XE=250  : YE=180
140 XB=0    : YB=0
150 XW=XE-XB : YW=YE-YB
160 XS=1
170 '
180 SCREEN 2 : COLOR 15.1,1 : CLS
190 FOR T=1 TO 20
200   X=INT(RND(1)*XW)+XB
210   Y=INT(RND(1)*YW)+YB
220   LINE (X,Y)-(X+10,Y+10),11,BF
230 NEXT T
240 '
250 X=0 : Y=INT(RND(1)*(YW+1))+YB
260 I$=INKEY$ : IF I$<>"" THEN END
270 IF X>XE THEN 250
280 IF POINT(X,Y)<>1 THEN BEEP : GOTO 250
290   PSET (X,Y),7
300   X=X+XS
310 GOTO 260
```

180〜230：的をランダムに表示する

250：ビームを描き始める位置を設定する

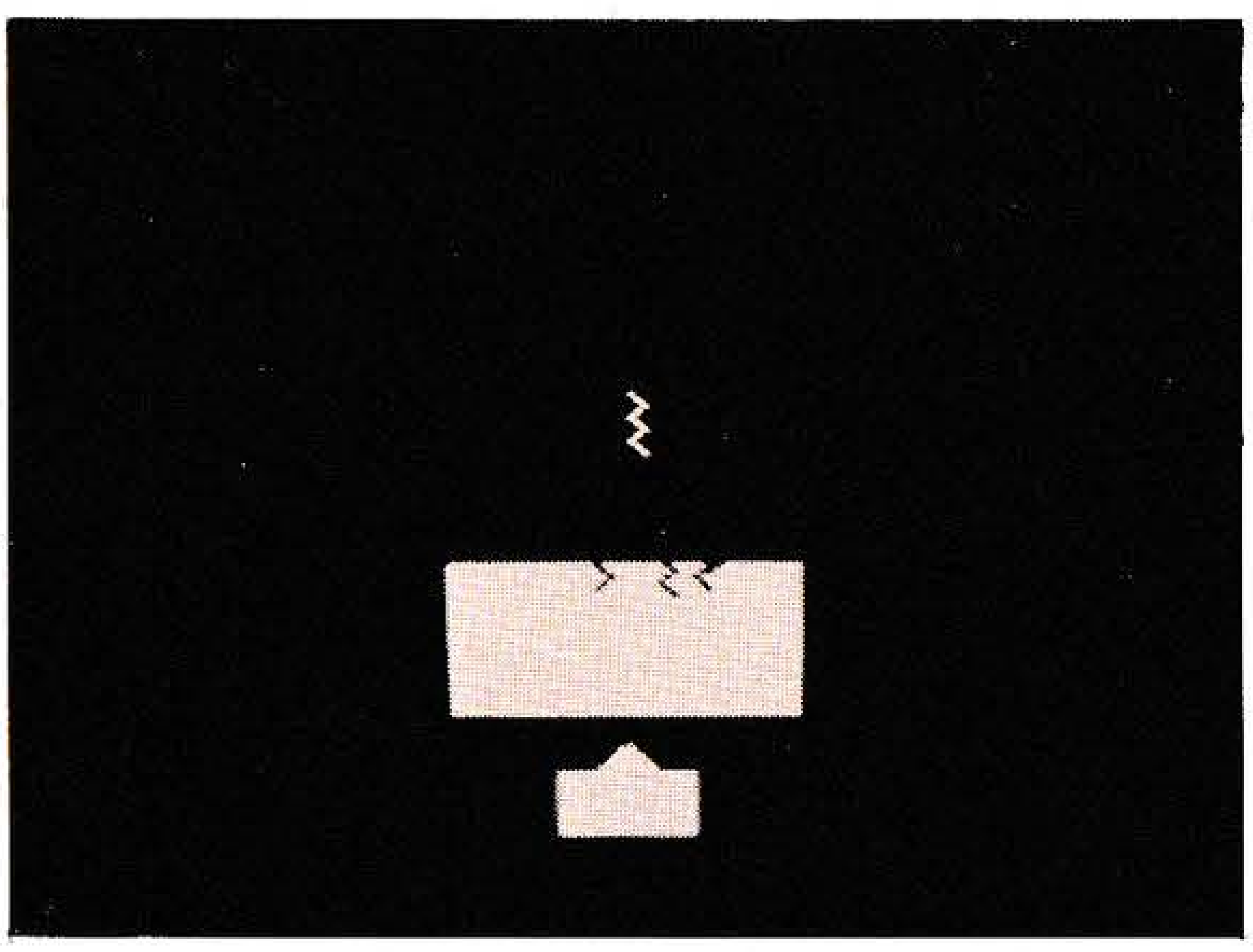
基地が危ない!

降りそそぐミサイル．防御用の壁が，目の前でみるみる破壊されていく．ミサイル基地が危ない!なんて場面がゲームにはよくありますが，いったいどうしたらこんな画面が描けるのでしょうか．シンプルなかたちでそれを再現してみたのがこのプログラムです．

ミサイルは，その先に物体があるかないかを調べながら降りそそいでいます．その判定には，やはりPOINT関数を使っています．このミサイルは，一度に縦8ドットずつ落ちていきます．そして，壁にミサイルが当たったときにくい込んだ分だけ壁は破壊されていきます（*Figure 1* 参照）．この破壊されてできた細長い穴がだんだん重なって，ついには壁をつらぬき，基地にミサイルが命中してしまうわけです．

ミサイルの動きは，250〜300行のループで行っています．降りそそいでくるミサイルは，*Figure 2* のような2種類のパターンを交互に表示しています．このパターンの形を決めているのが，FGという変数です．この変数は，290行のようにミサイルが1ステップ（8ドット）分落ちるたびに1，−1を繰り返し，このFGの値により330行でパターンを選び，DRAW命令を用いて表示します．これが降りそそぐイナズマ・ミサイルの実体です．

ここで330行のA$を見てください．このA$は，イナズマ・ミサイルをDRAW命令のグラフィックマクロ命令で表したものです．このA$をいろいろ変えてみると，おもしろい形のミサイルができると思います．ためしてみますか!?



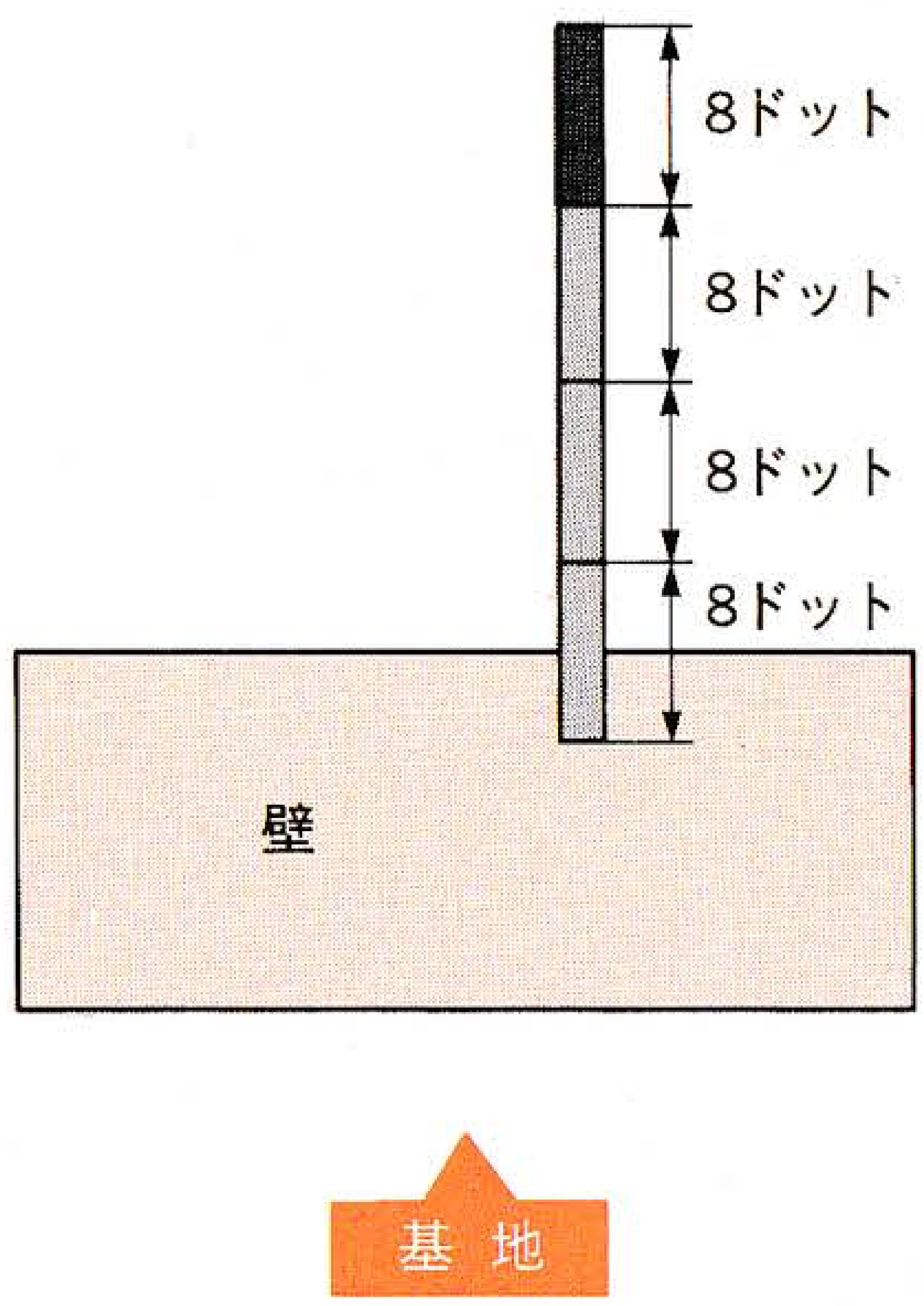


Figure 1

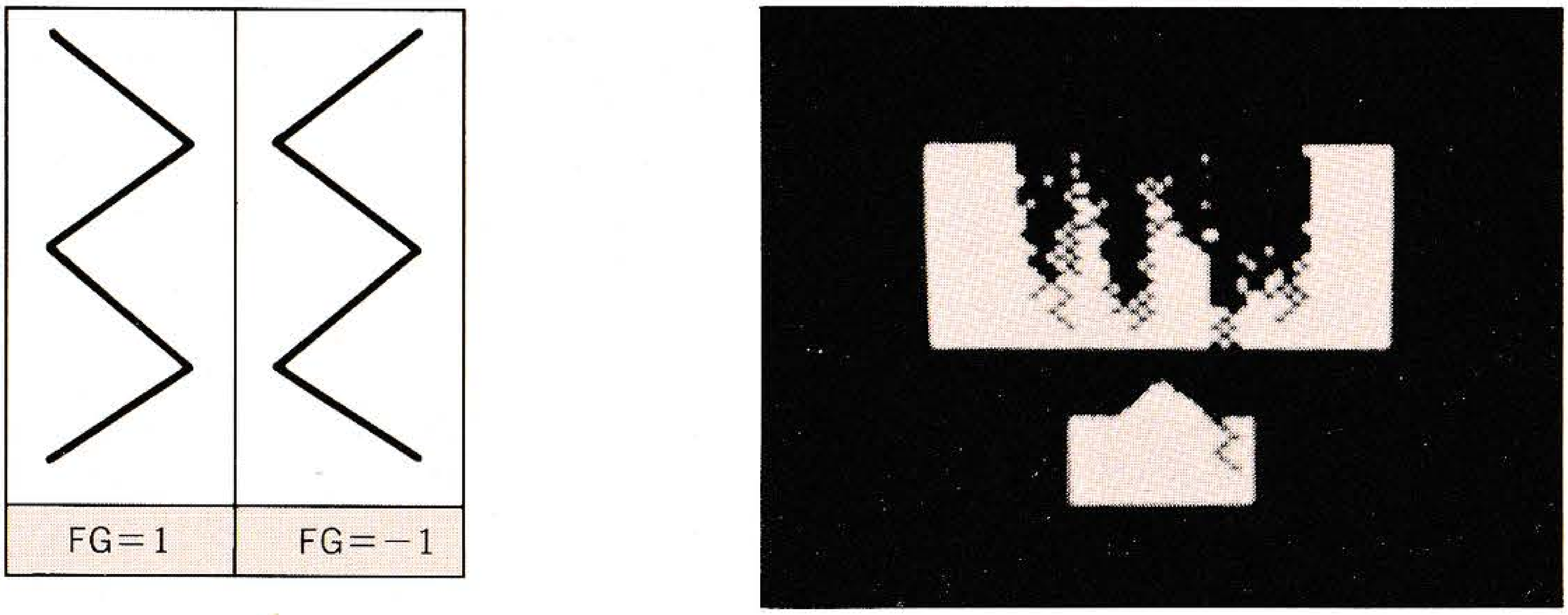


Figure 2

List 4

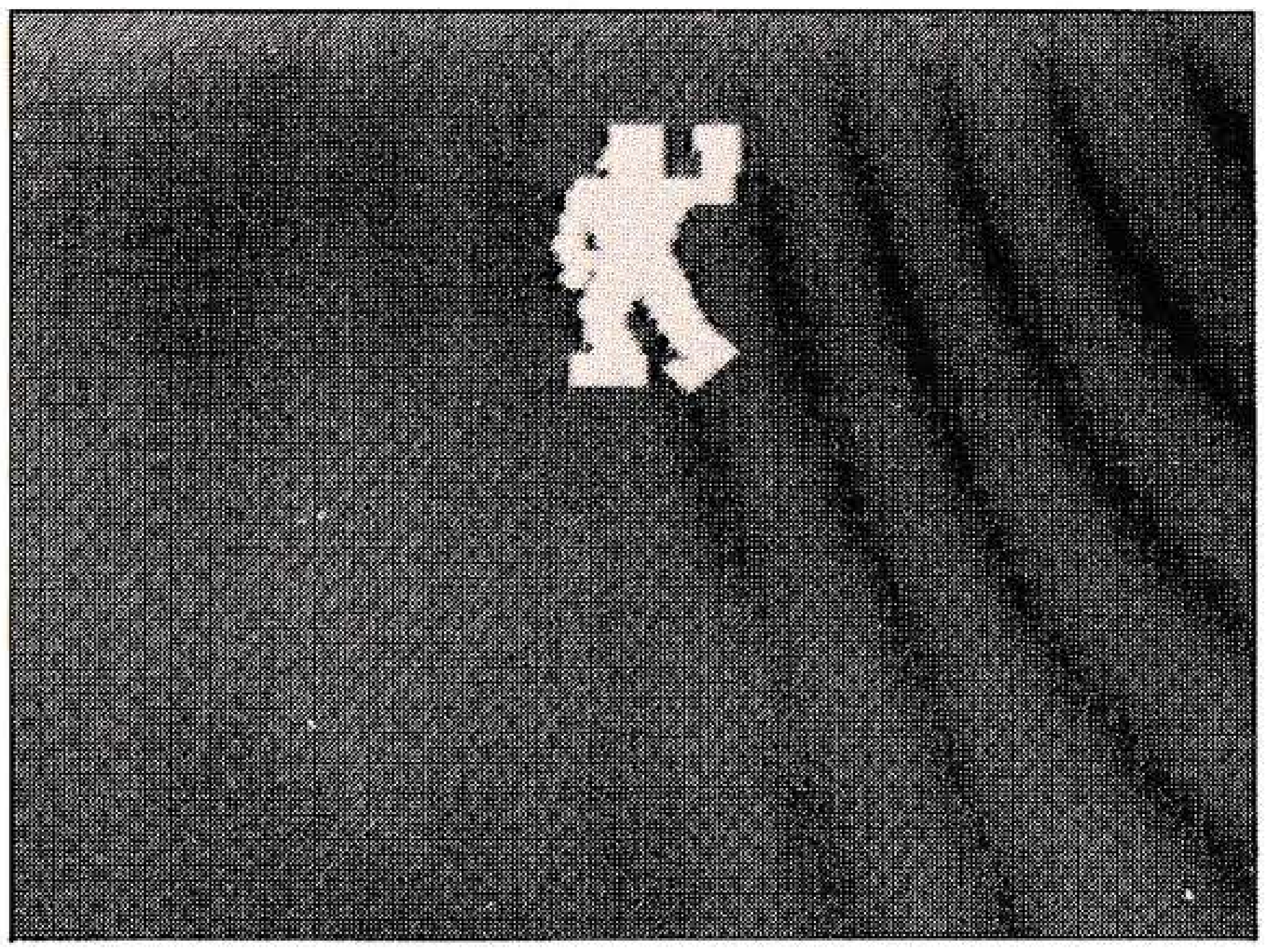
```
100 ' List   4
110 '
120 YS=8
130 FG=1
140 C1=11 : C2=7
150 FC=15 : BC=0
160 DM=RND(-TIME)
170 '
180 SCREEN 2 : COLOR FC,BC,BC : CLS
190 LINE (100,145)-(150,170),C1,BF
200 DRAW "S4A0 BM125,175C=C2;M130,180M135,180M135,190M115,190"
210 DRAW "M115,180M120,180M125,175" : PAINT STEP(0,+5),C2
220 '
230 X=INT(RND(1)*(140-110))+110
240 Y=INT(RND(1)*8)+10
250 P=POINT(X,Y+1) : IF P<>BC THEN 370
260   Y=Y+YS
270   C=FC : GOSUB 320
280   C=BC : GOSUB 320
290   FG=-FG
300 GOTO 250
310 '
320 D$="BM=X;,=Y;"
330 IF FG=1 THEN A$="C=C;E2H2E2H2E2" ELSE A$="BR2C=C;H2E2H2E2H2"
340 DRAW D$+A$
350 RETURN
360 '
370 IF P<>C2 THEN PLAY "T255L6407A" : GOTO 230
380 PLAY "T255L1002A":DM$=INPUT$(1):END
```

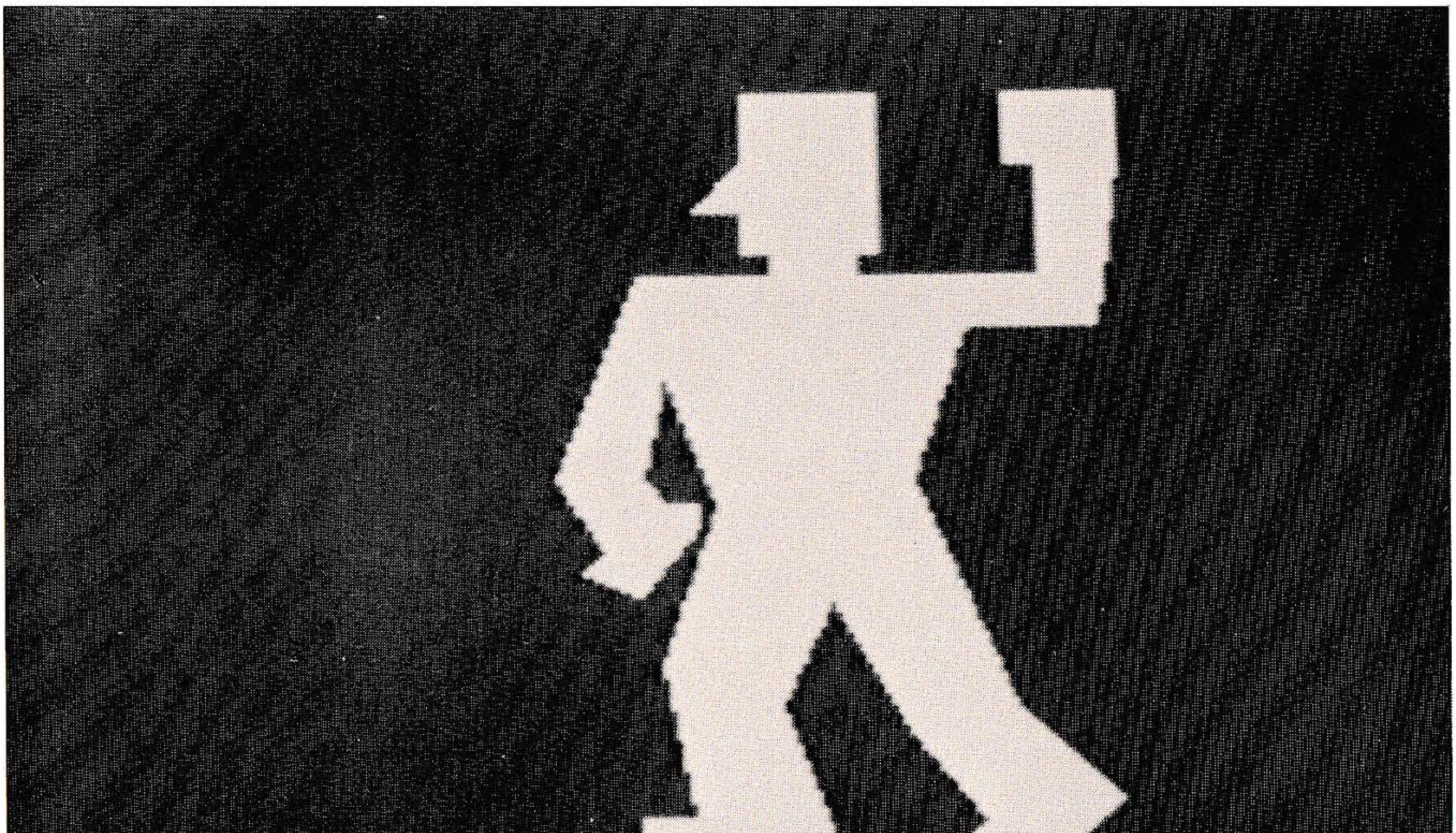
（140〜150）カラーコード　0…透明　11…明るい黄　7…水色　15…白

（250〜300）メインルーチン

図形の拡大，縮小

ある図形を描くときに，それをもう少し大きくしたいとか，もっと小さく描けたらいいのにと思うことはよくあることです．そこで，図形の拡大・縮小について考えてみましょう．図形の拡大・縮小は，相対座標を用いると簡単に実現できます．たとえば，*Figure 1* のように，ある四角形を考えると，この四角形の各辺を2倍にするには，各頂点の座標に2を掛ければよいのです．





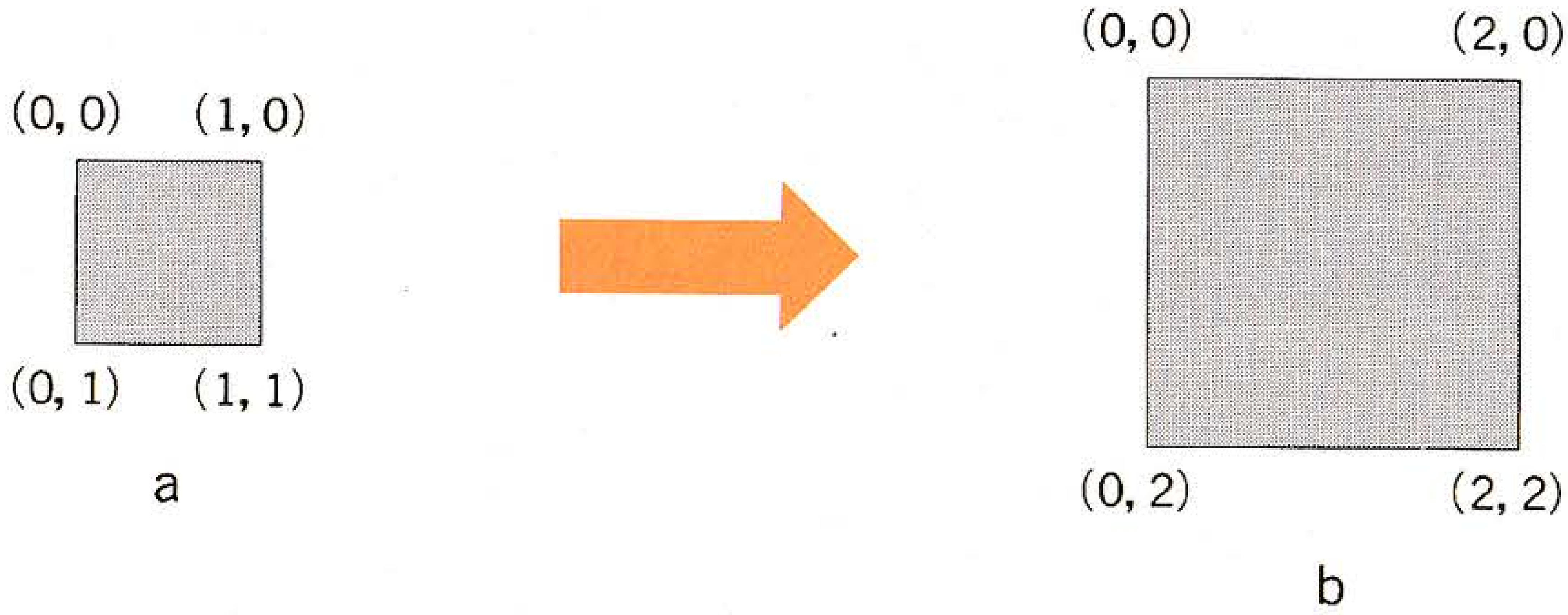
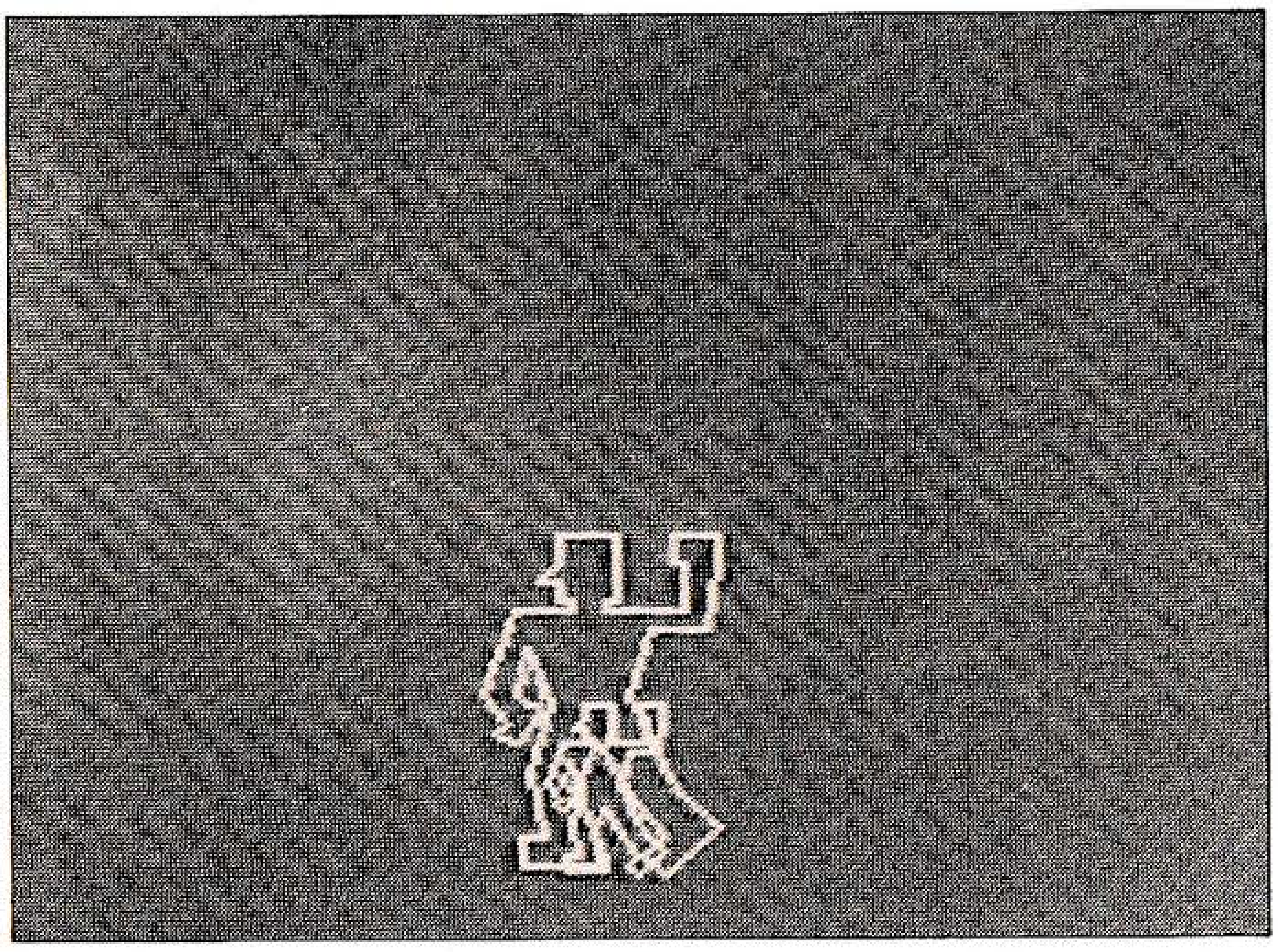


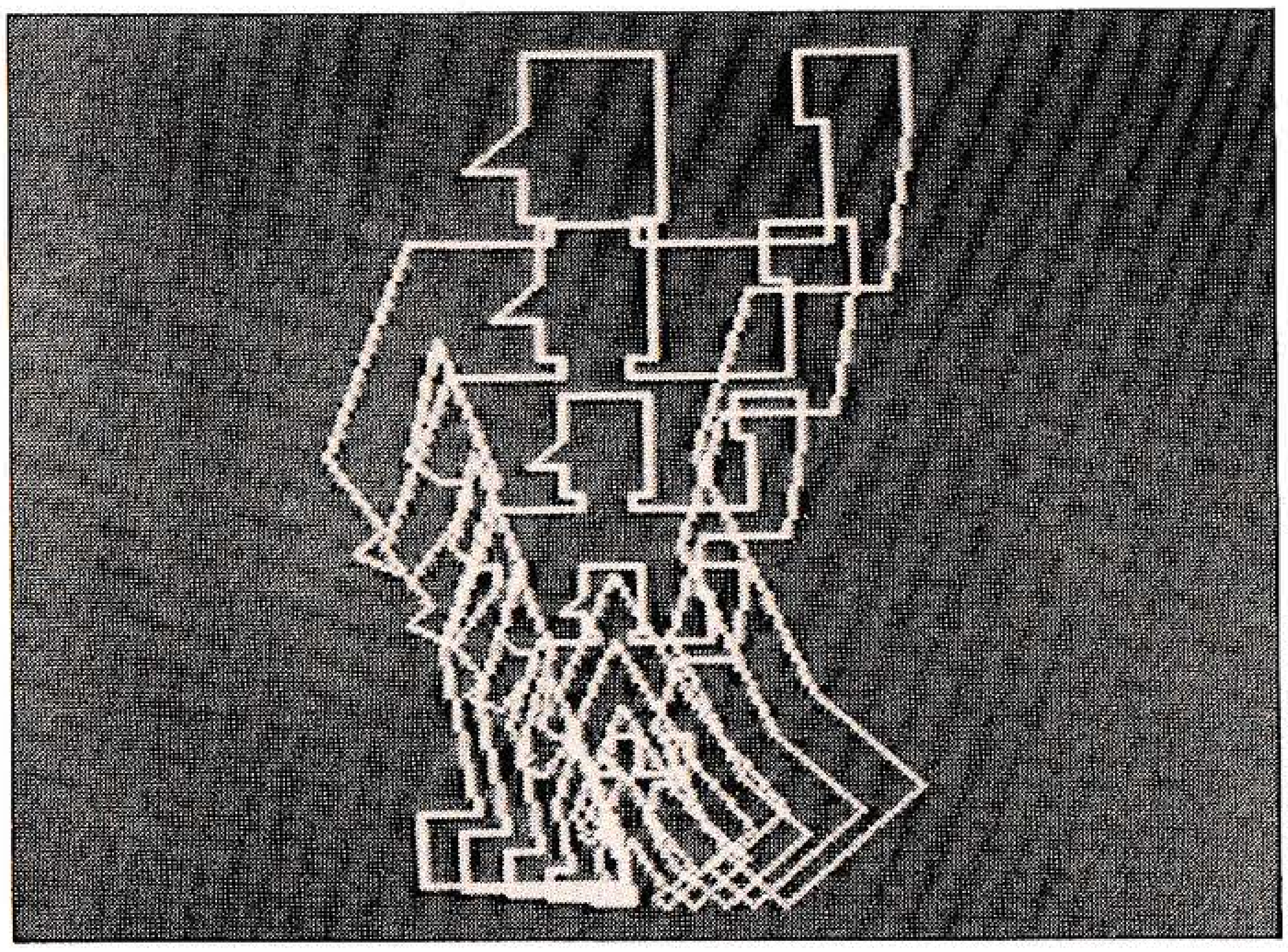
Figure 1

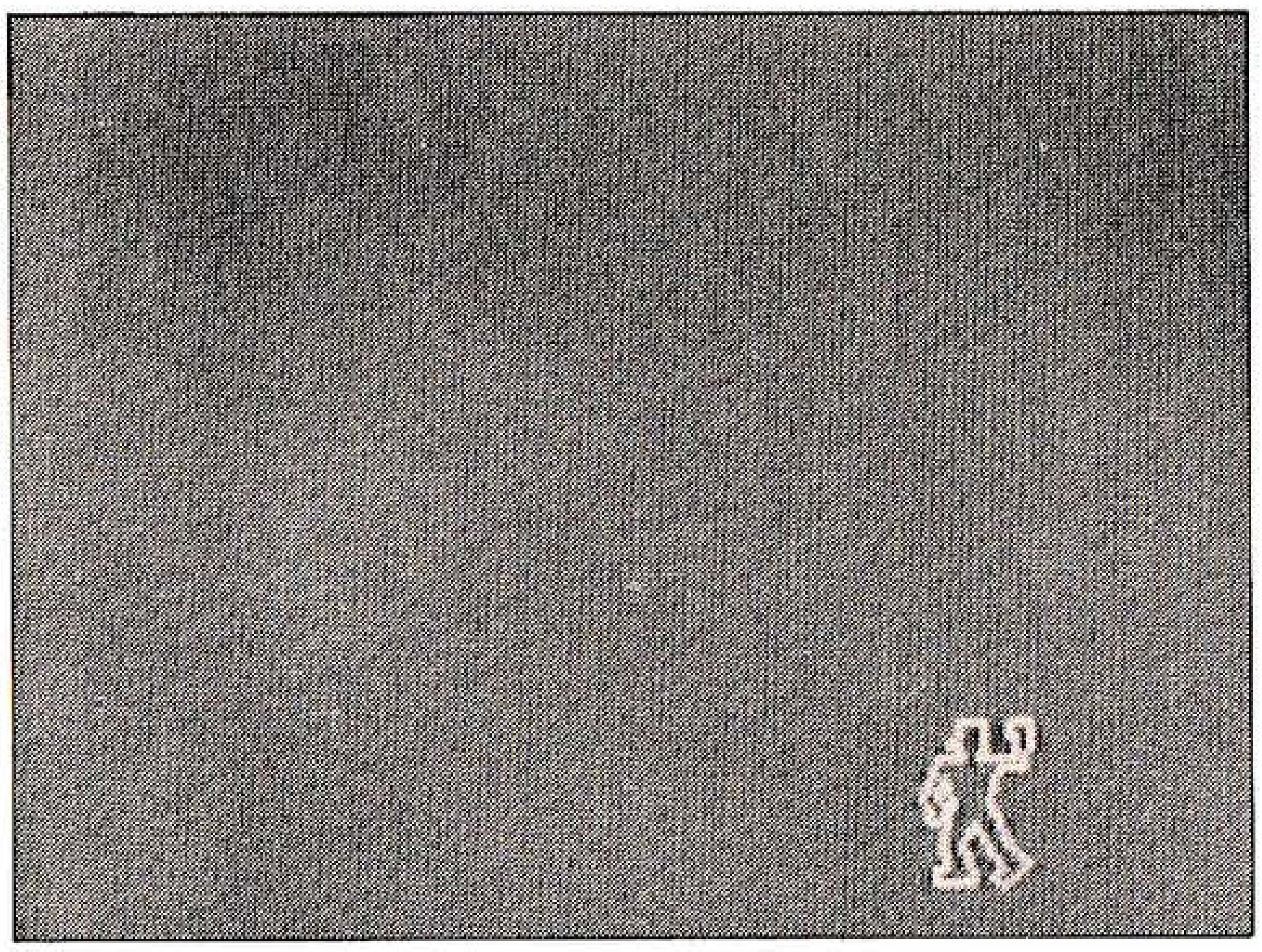
List 5a

```
100 ' List  5a
110 '
120 X0=100 : YS=2 : C=15
130 '
140 DIM X(50),Y(50)
150 RESTORE 370
160 FOR I=1 TO 48                  データを配列に読み込む
170   READ X(I),Y(I)
180 NEXT I
190 '
200 SCREEN 2 : COLOR 15,13,13 : CLS
210 Y=0
220 FOR M=.4 TO 4.4 STEP .8
230   X=X0 : Y=Y+YS : GOSUB 290
240   FOR T=1 TO 100 : NEXT T                   メインルーチン
250   LINE (X,Y)-(X+25*M,Y+40*M),13,BF
260 NEXT M
270 END
280 '
290 T=15-M*3 : PLAY "T80L4O3N=T;"
300 PSET (X+X(1)*M,Y+Y(1)*M),C
310 FOR I=2 TO 48
320   LINE -(X+X(I)*M,Y+Y(I)*M),C               人を描くサブルーチン
330 NEXT I
340 PAINT (X+12*M,Y+20*M),C
350 RETURN
360 '
370 DATA 7,0, 12,0, 12,7, 11,7, 11,8, 18,8, 18,3, 17,3
380 DATA 17,0, 21,0, 21,3, 20,10, 15,10, 13,17, 17,27, 21,31
390 DATA 16,36, 14,34, 16,32, 13,28, 10,22, 8,27, 10,33, 10,35
400 DATA 10,33, 10,35, 3,35, 3,32, 5,32, 4,25, 6,18, 4,12
410 DATA 3,17, 4,18, 5,18, 5,19, 3,22, 1,21, 2,20, 0,17
420 DATA 3,8, 8,8, 8,7, 7,7, 7,5, 5,5, 7,3, 7,0
```









それでは，*List 5a* を見てください．

370行以降の DATA は，相対座標を用いて表した図形の座標データです．今までのプログラムでは，このデータをただ単に始点の座標と足して出力していました．しかし，ここでは図形の拡大をするため，読み取ったデータに，倍率の変数 M を掛けてから出力します．この変数を1より大きくすれば図形は拡大され，1より小さくすれば縮小されます．

このプログラムでは，人物が徐々に拡大しながら近づいてきます．このとき，拡大しながら移動する人物を消しているのは，250行の LINE 命令です．

拡大，縮小といえば，DRAW 命令のスケーリングファクタを利用することもできます．DRAW 命令の文字列が，U，D，L，R，E，F，G，H および相対座標指定の M により設定されているとき，S を設定することでその大きさを変えることができる，というものです．このスケーリングファクタを利用したプログラムを *List 5b* に示しました．

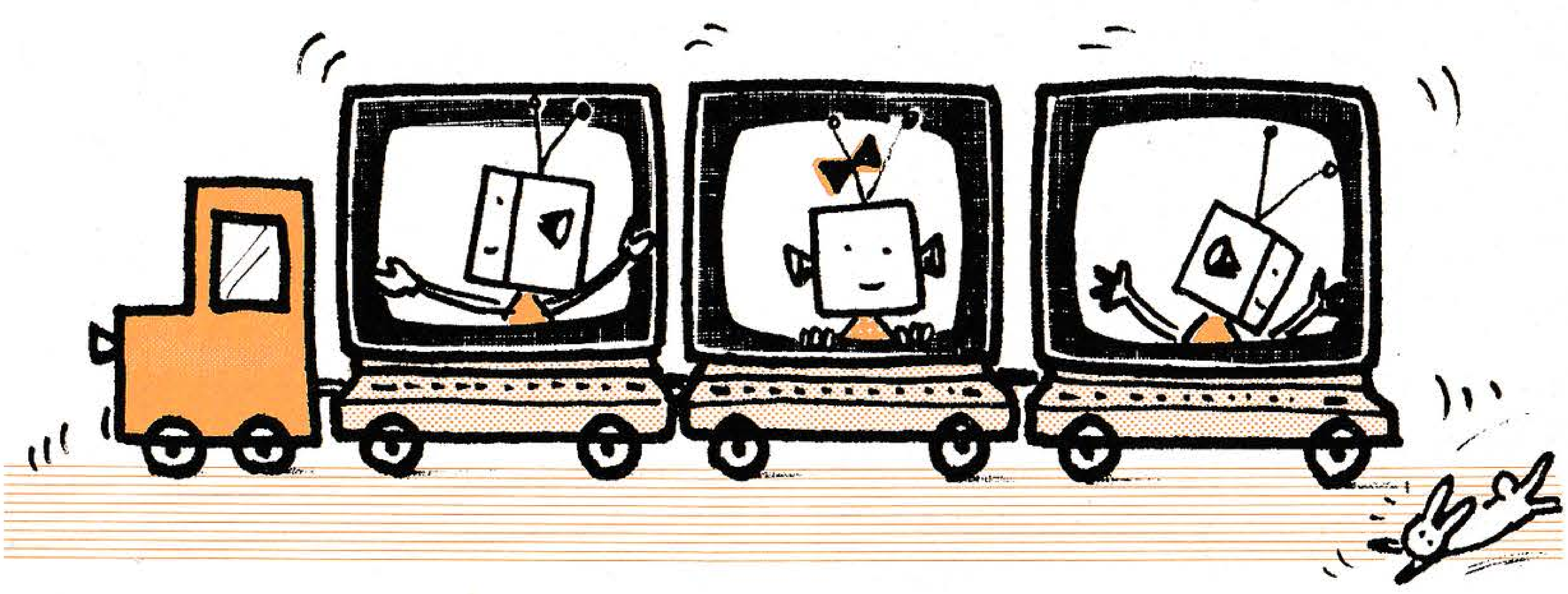
List 5b

```
100 ' List   5b
110 '
120 SCREEN 2 : COLOR 15,4,4 : CLS
130 DEFINT A-Z
140 FOR X=200 TO 120 STEP -10
150   X0=X : Y0=150 : S=4
160   C=15 : GOSUB 310
170   PLAY "T255L32O2A"
180   FOR T=1 TO 300 : NEXT T
190   C=4  : GOSUB 310
200 NEXT X
210 '
220 FOR I=0 TO 16 STEP 4
230   X0=110-I*21/8
240   Y0=150-I*9
250   S=I+4 : PLAY "L64CC+DD+EFGG+AA+B"
260   C=15  : GOSUB 310
270   FOR T=1 TO 300 : NEXT T
280 NEXT I
290 DM$=INPUT$(1):END
300 '
310 DRAW "S4A0 BM=X0;,=Y0; S=S; C=C;"
320 DRAW "BM+7,+0M+5,+0M+0,+7M-1,+0M+0,+1M+7,+0M+0,-5M-1,+0"
330 DRAW "M+0,-3M+4,+0M+0,+3M-1,+7M-5,+0M-2,+7M+4,+10M+4,+4"
340 DRAW "M-5,+5M-2,-2M+2,-2M-3,-4M-3,-6M-2,+5M+2,+6M+0,+2"
350 DRAW "M-7,+0M+0,-3M+2,+0M-1,-7M+2,-7M-2,-6M-1,+5M+1,+1"
360 DRAW "M+1,+0M+0,+1M-2,+3M-2,-1M+1,-1M-2,-3M+3,-9M+5,+0"
370 DRAW "M+0,-1M-1,+0M+0,-2M-2,+0M+2,-2M+0,-3"
380 RETURN
```

人を移動するルーチン (lines 140–200)

人を拡大するルーチン (lines 220–290)

人を描くサブルーチン (lines 310–380)



ストップウォッチを作る

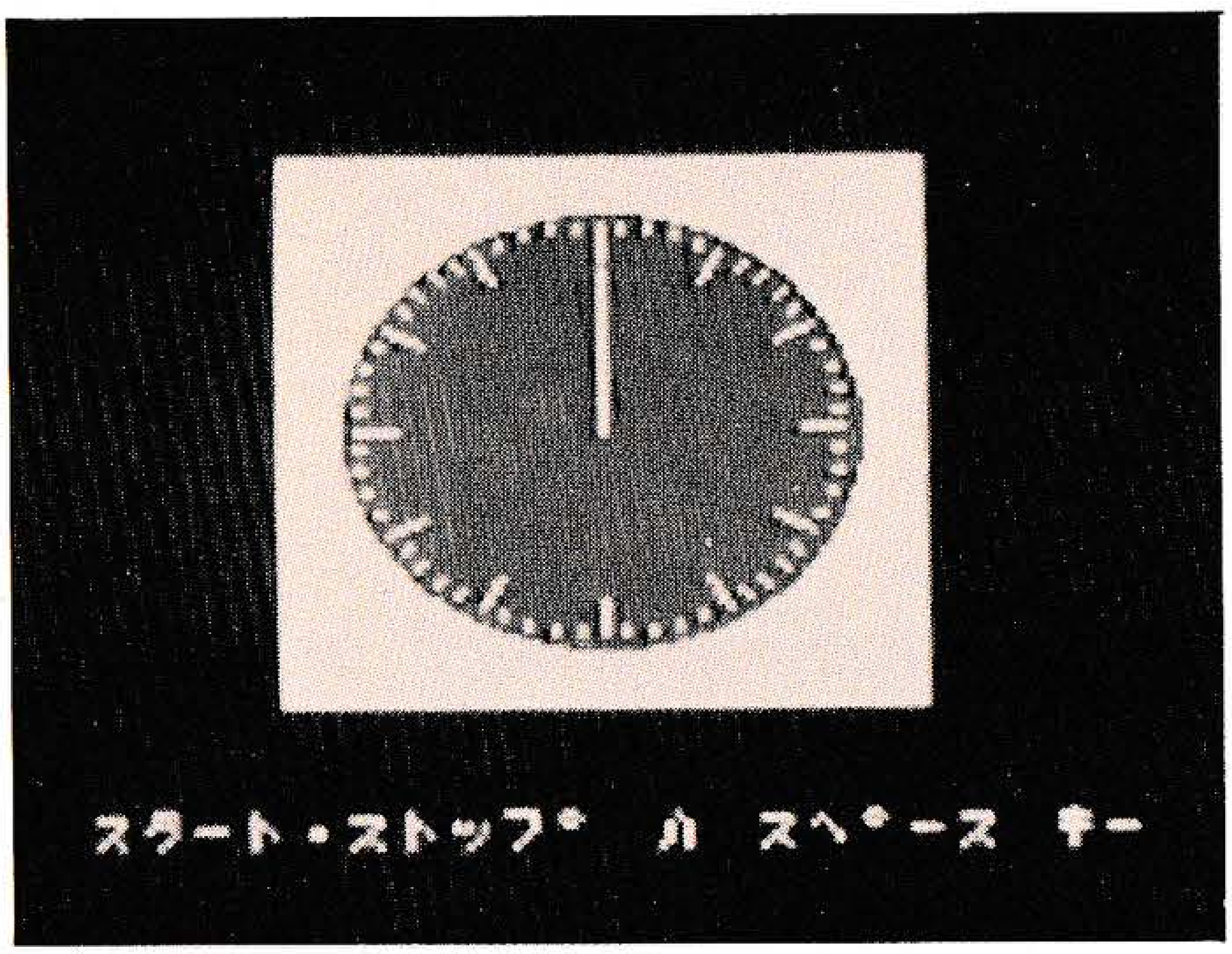
MSX の内部には，インターバルタイマーと呼ばれる時計があり．一定の時間ごとにプログラムに割り込みをかけることができます．この機能を利用して，ストップウォッチを作ってみましょう．

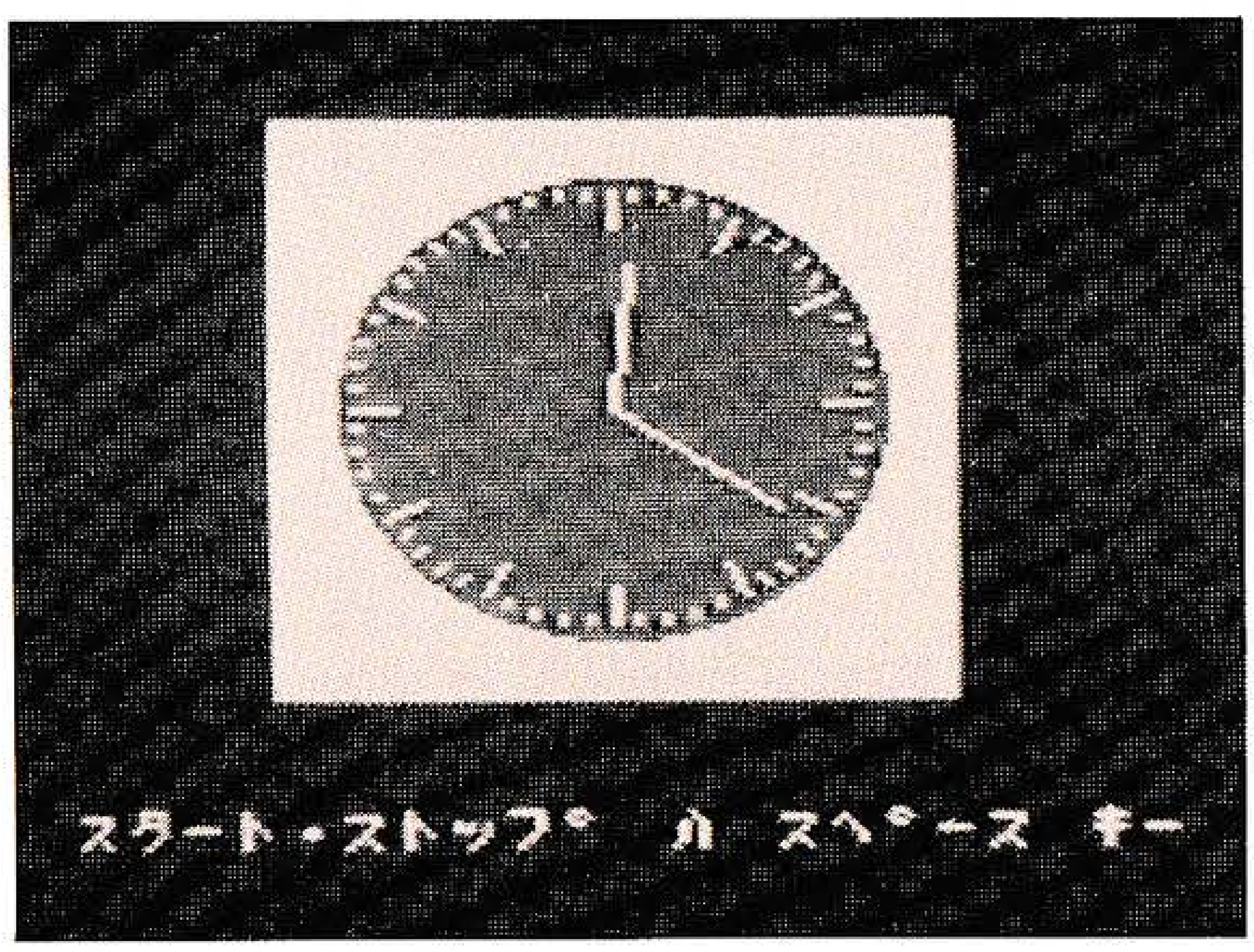
まず160，170行で時計の枠を描き，180〜250行で目盛りを書き込んでいます．右まわりに中心の角度を6度ごとに増やしていき，目盛りの座標を求めていくわけですが，そのとき算出したサイン，コサインの関数値は，後で利用するため配列 S（I）と配列 C（I）に格納しておきます．三角関数を一回ごとに計算するよりも，配列変数の値を調べるほうが時間がかからないからです．これはプログラムのスピードアップ手段として，よく使われるテクニックです．

300行では60／60秒に1回，すなわち1秒ごとに割り込みがかかるよう，タイマーの周期を決めています．また，時計のスタート・ストップはスペースバーによる割り込みで制御することにして，その設定を310行で行っています．

以上の設定が終わると，プログラムは330行の無限ループに入ります．そしてタイマー割り込みが発生すると350行のサブルーチンが実行され，1秒進めて分針と秒針を書き直します．

一方スペースバーを押すと，470行の割り込みサブルーチンが実行されます．そのたびに変数 F の値が1，2，3，1，2，3……と変わって，針をスタートさせるか，ストップさせるか，もとの位置に戻すかを決めています．





List 6

```
100 ' List   6
110 '
120 X0=128 : Y0=90 : DG=3.14159/180
130 DIM S(59),C(59)
140 '
150 SCREEN 2 : COLOR 15,2,2 : CLS
160 LINE (X0-50,Y0-50)-(X0+50,Y0+50),15,BF
170 CIRCLE (X0,Y0),40,4 : PAINT(X0,Y0),4
180 RL=38
190 FOR I=0 TO 59
200   T=I*6*DG : S(I)=SIN(T) : C(I)=COS(T)
210   IF (I MOD 5)=0 THEN RS=32 ELSE RS=37
220   X1=X0+RL*S(I) : Y1=Y0-RL*C(I)
230   X2=X0+RS*S(I) : Y2=Y0-RS*C(I)
240   LINE (X1,Y1)-(X2,Y2),15
250 NEXT I
260 M=0 : S=0 : C=15 : GOSUB 410
270 OPEN "GRP:" FOR OUTPUT AS #1
280 DRAW "BM48,160" : PRINT #1,"ｽﾀｰﾄ･ｽﾄｯﾌﾟ ﾊ ｽﾍﾟｰｽ ｷｰ"
290 '
300 ON INTERVAL=60 GOSUB 350
310 ON STRIG GOSUB 470 : F=0 : STRIG(0) ON
320 '
330 GOTO 330
340 '
350 C=4  : GOSUB 410
360 S=S+1 : IF S=60 THEN S=0 : M=M+1 : IF M=60 THEN M=0
370 C=15 : GOSUB 410
380 PLAY "V14L64O6A"
390 RETURN
400 '
410 X=X0+25*S(M) : Y=Y0-25*C(M)
420 LINE (X0,Y0)-(X,Y),C
430 X=X0+30*S(S) : Y=Y0-30*C(S)
440 LINE (X0,Y0)-(X,Y),C
450 RETURN
460 '
470 F=F+1 : IF F=4 THEN F=1
480 ON F GOTO 490,500,510
490 INTERVAL ON  : RETURN
500 INTERVAL OFF : RETURN
510 C=4 : GOSUB 410
520 M=0 : S=0 : C=15 : GOSUB 410
530 RETURN
```

定数の設定 (120)

ストップウォッチの枠を描く (160–170)

ストップウォッチの目盛を描く (190–250)

針を描くサブルーチン (410–440)

迷路を作る

誰でも，紙の上に書かれた迷路を鉛筆の先でたどっていく，という経験が，一度や二度はあると思います．

この迷路をコンピュータに作らせるにはどうすればよいのでしょうか．

迷路を作るには，道をのばしていく方法と，壁をのばしていく方法がありますが，ここでは壁をのばしていく方法を使います．*Figure 1* のように入口と出口によって，外壁は2つに分けられます．この外壁(壁a，b)に接する壁を次々に作っていくのですが，すでに存在している壁と壁をつなぐような壁を作ってはいけません．抜け道がふさがってしまうことがあるからです．この原則さえ守って壁を作っていけば，かならず抜け道のある迷路ができあがります．

このプログラムでは，220行で外壁を描き，230，240行で入口と出口を作っています．

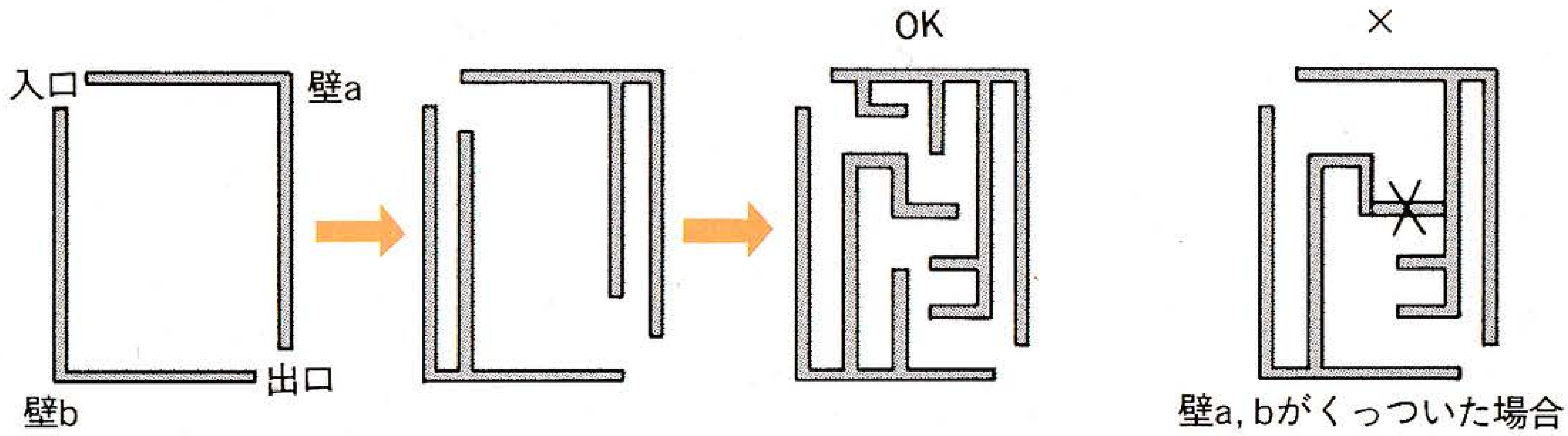
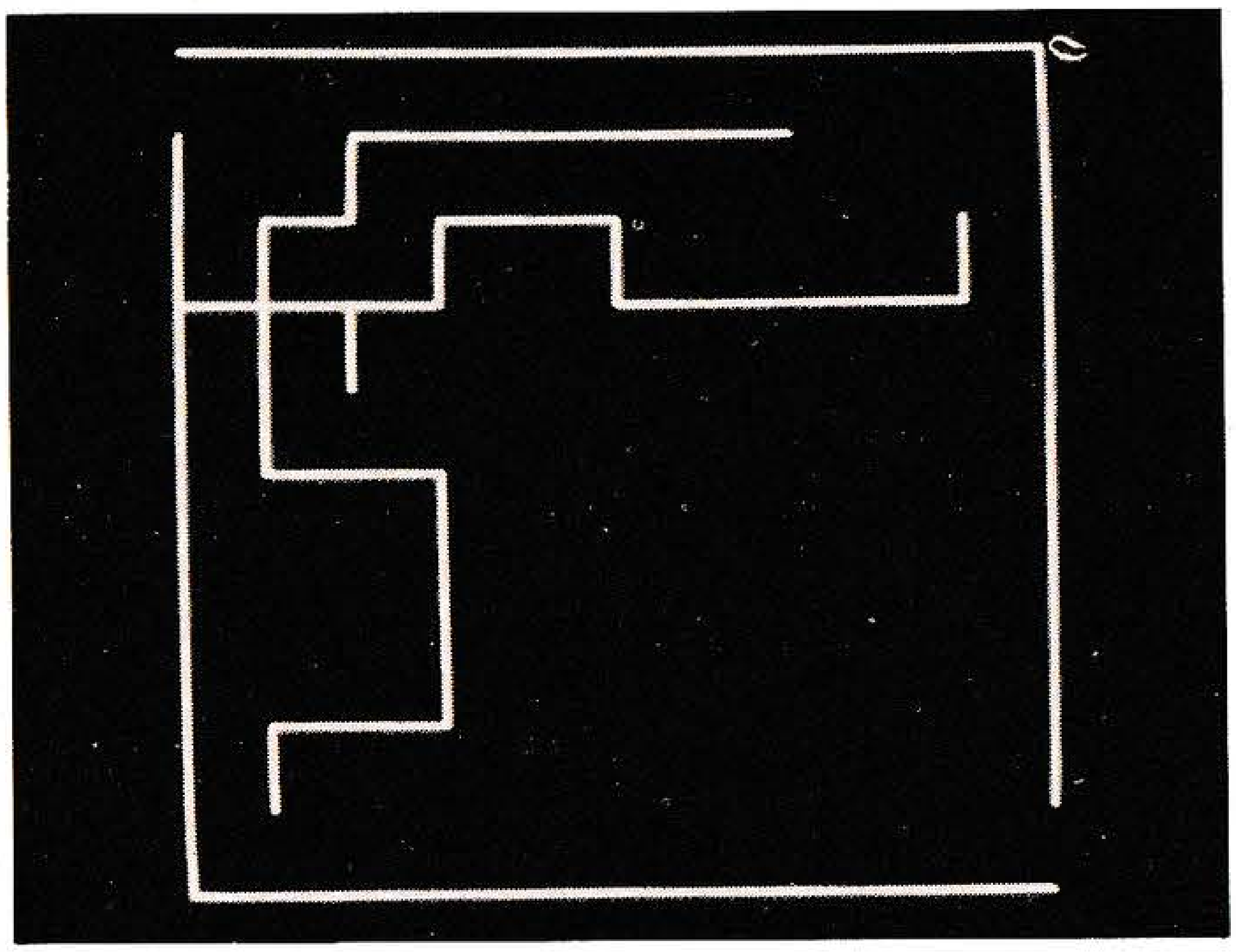
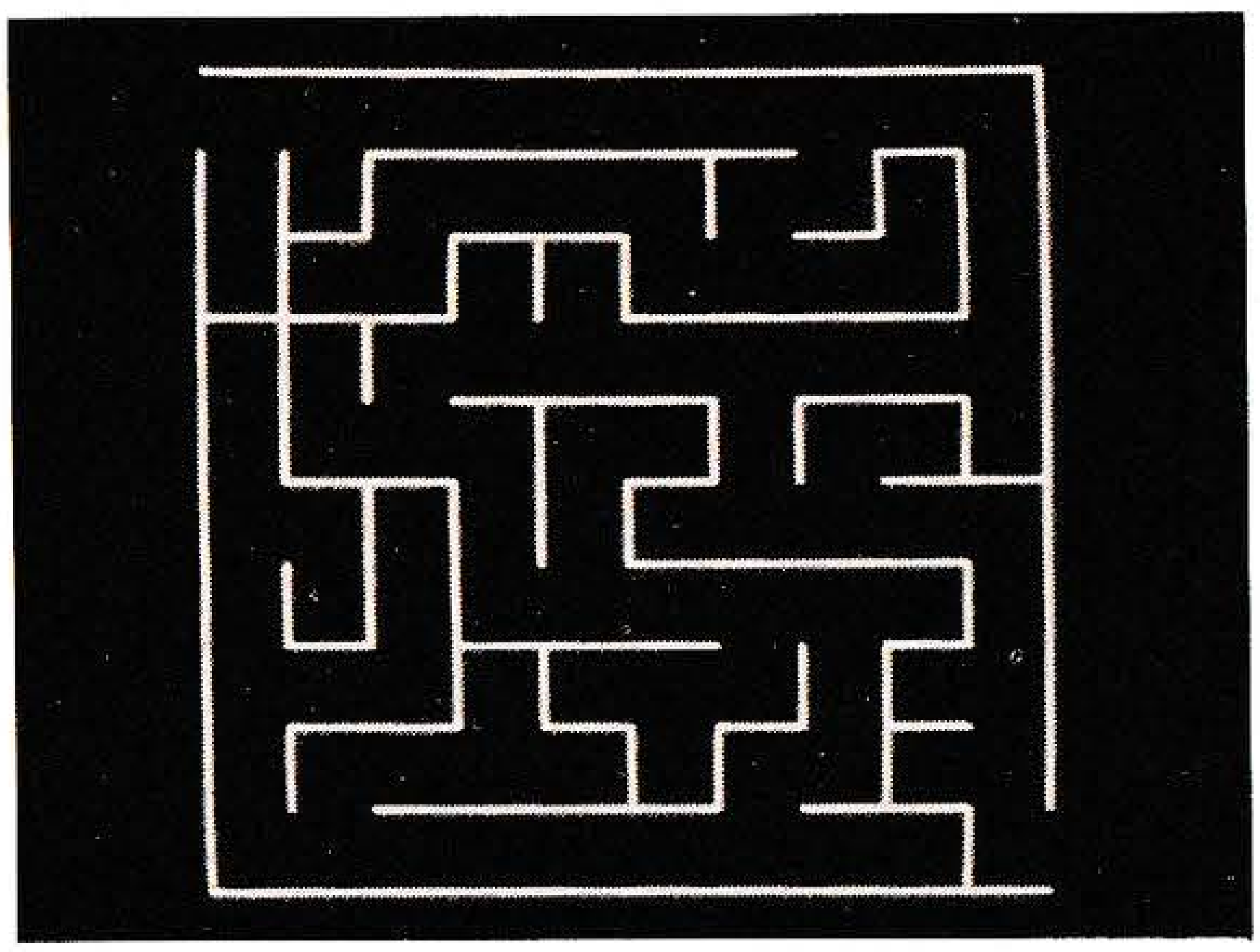


Figure 1





260〜280行は，2つの壁のどちらか一方に接する点を探しています．その点が決まると，300行で発生させた乱数により，どちら向きの壁を作るか決定し，310〜380行でその壁が他の壁と接しないかどうか判定し，390行で壁を描きます．

この方法は，大きい迷路を作るとき非常に時間がかかるのが欠点です．他にも迷路の描き方はありますから，研究してみてください．

List 7

```
100 ' List   7
110 '
120 DEFSNG A-Z
130 XB=50 : XE=200
140 YB=10 : YE=180
150 M=10  : N=10
160 XW=(XE-XB)/M : YW=(YE-YB)/N
170 S=(M-1)*(N-1)
180 DM=RND(-TIME)
190 C=7
200 '
210 SCREEN 2 : COLOR 15,1,1 : CLS
220 LINE (XB,YB)-(XE,YE),C,B
230 LINE (XB,YB+1)-(XB,YB+YW-1),1
240 LINE (XE,YE-1)-(XE,YE-YW+1),1
250 '
260 X=INT(RND(1)*(M+1)) : Y=INT(RND(1)*(N+1))
270   XO=XB+X*XW : YO=YB+Y*YW
280   IF POINT(XO,YO)=1 THEN 260
290   FOR I=1 TO 20
300     R=INT(RND(1)*4)
310     IF R=0 THEN XS=-1 : YS=0 : GOTO 350
320     IF R=1 THEN XS=1  : YS=0 : GOTO 350
330     IF R=2 THEN XS=0  : YS=1 : GOTO 350
340     IF R=3 THEN XS=0  : YS=-1
350     XN=XO+XS*XW : YN=YO+YS*YW
360     IF XN<=XB OR XN>=XE THEN 410
370     IF YN<=YB OR YN>=YE THEN 410
380     IF POINT(XN,YN)<>1 THEN 410
390     LINE (XO,YO)-(XN,YN),C
400     XO=XN : YO=YN : S=S-1
410   NEXT I
420 IF S>0 THEN 260
430 PLAY "V12T240L64O5A"
440 DM$=INPUT$(1):END
```

220行 ……外壁を描く
230〜240行 出入口を描く
300行 ……壁を延ばす方向を決める
310〜380行 新たに作る壁が他の壁に接しないかどうか判定する

音符とともに音楽を！

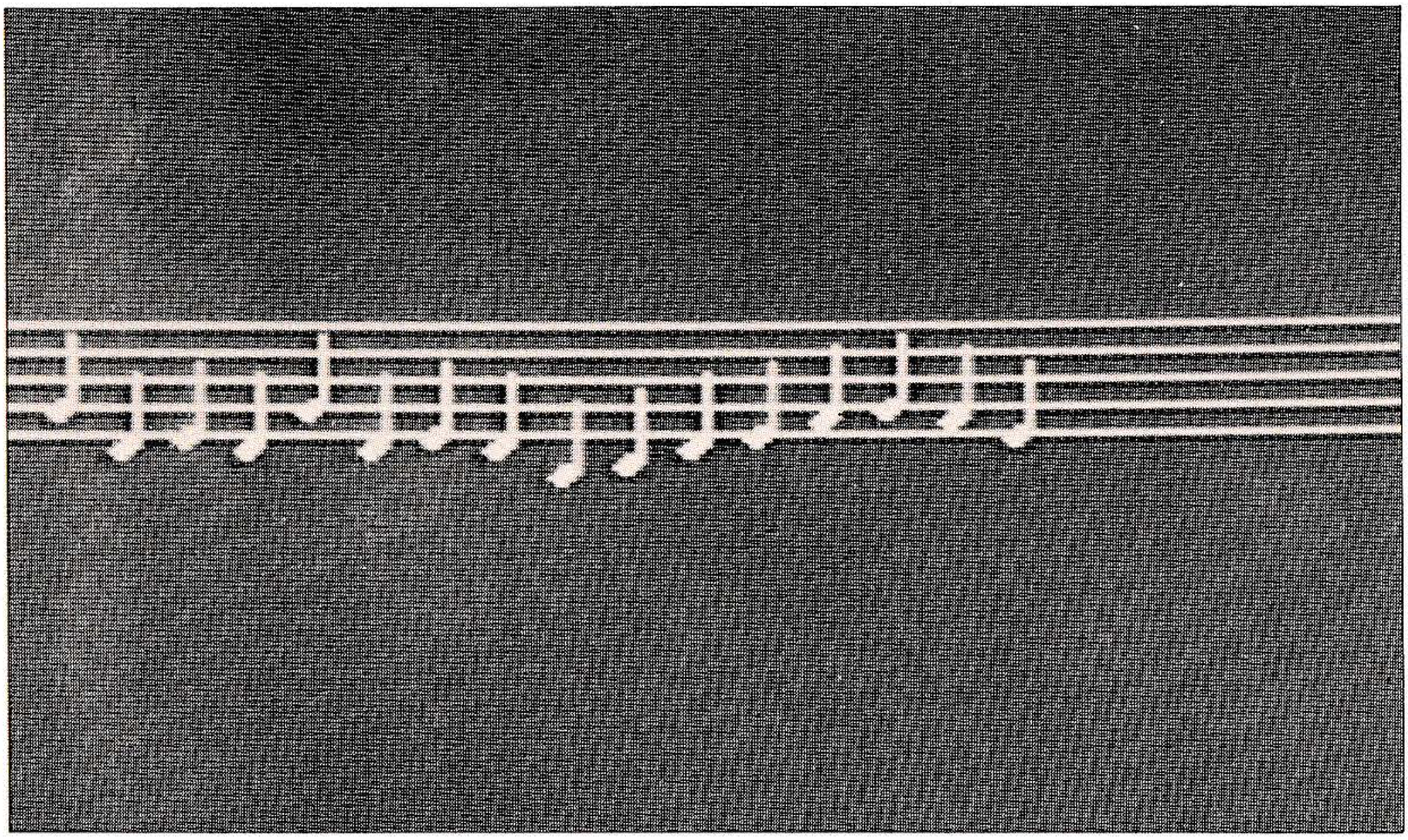
MSXは，PLAY命令を使うことによって，音楽を演奏することができます．そこで，この音楽とともに，画面に音符を出力するプログラムを考えてみましょう．プログラムを簡単にするために，単音で音符を出力します．

音符の形（♩）は，DRAW命令で毎回描いています．

では，リストを見てください．音符は1画面に24個しか並べられないので，それを変数Nで数えておき，N=25になったら画面をクリアして再び左端から音符を出力しています．

音のデータをDATAとして蓄えておく方法はいろいろあると思いますが，ここでは，1つひとつをカンマで区切って，大文字だけで表現させています．また，音程のデータである"C，D，E……"が，ドである"C"から数えて何番目の高さにあるかということは，130行で文字列 TN$="CDEFGAB"と定義し，270行でINSTR関数によって得ています．この値を使って280行の式で計算し，音符をDRAW命令で出力する場所を決めているのです．

このプログラムでは，最も単純な場合を例にとりましたが，MSXは3重和音まで出



せるので，それを音符とともに出力するようなプログラムを考えてみてください．また，休符や，#，♭なども出力するとより楽しいものとなるでしょう．

List 8

```
100 ' List   8
110 '
120 C=15
130 TN$="CDEFGAB"
140 PLAY "S0M5000T80"
150 '
160 SCREEN 2 : COLOR 15,4,4 : CLS
170 GOSUB 330
180 RESTORE 420
190 N=1
200 '
210 READ A$
220   IF A$="@" THEN DM$=INPUT$(1):END
230   B$=LEFT$(A$,1)
240   IF B$="O" THEN PLAY A$ : O=VAL(RIGHT$(A$,1)) : GOTO 210
250   IF B$="L" THEN PLAY A$ : GOTO 210
260   IF B$="R" THEN PLAY A$ : GOTO 210
270   I=INSTR(TN$,A$)
280   X=N*10  : Y=105-(O-4)*17.5-(I-1)*2.5
290   PLAY A$ : GOSUB 380
300   N=N+1 : IF N=25 THEN CLS : N=1 : GOSUB 330
310 GOTO 210
320 '
330 FOR L=80 TO 100 STEP 5
340   LINE (0,L)-(255,L),15
350 NEXT L
360 RETURN
370 '
380 DRAW "S4A0BM=X;,=Y;BM+2,+2C15"
390 DRAW "L1H1R3E1L3E1R2U12"
400 RETURN
410 '
420 DATA L16,O4,G,D,E,D,G,D,E,D
430 DATA L32,O3,B,O4,C,D,E,F,G,F,E
440 DATA F,E,D,E,D,C,O3,B,O4,C
450 DATA O3,B,O4,C,D,E,F,G,L16,O3,G
460 DATA L16,O4,G,D,E,D
470 DATA L32,O3,B,O4,C,D,E,F,G,F,E
480 DATA F,E,D,E,D,C,O3,B,O4,C
490 DATA O3,B,O4,C,D,E,F,G,L4,O3,G
500 DATA @
```

五線を描くサブルーチン（330～360）

音符を描くサブルーチン（380～400）

音のデータ（420～500）

飛行機を飛ばす

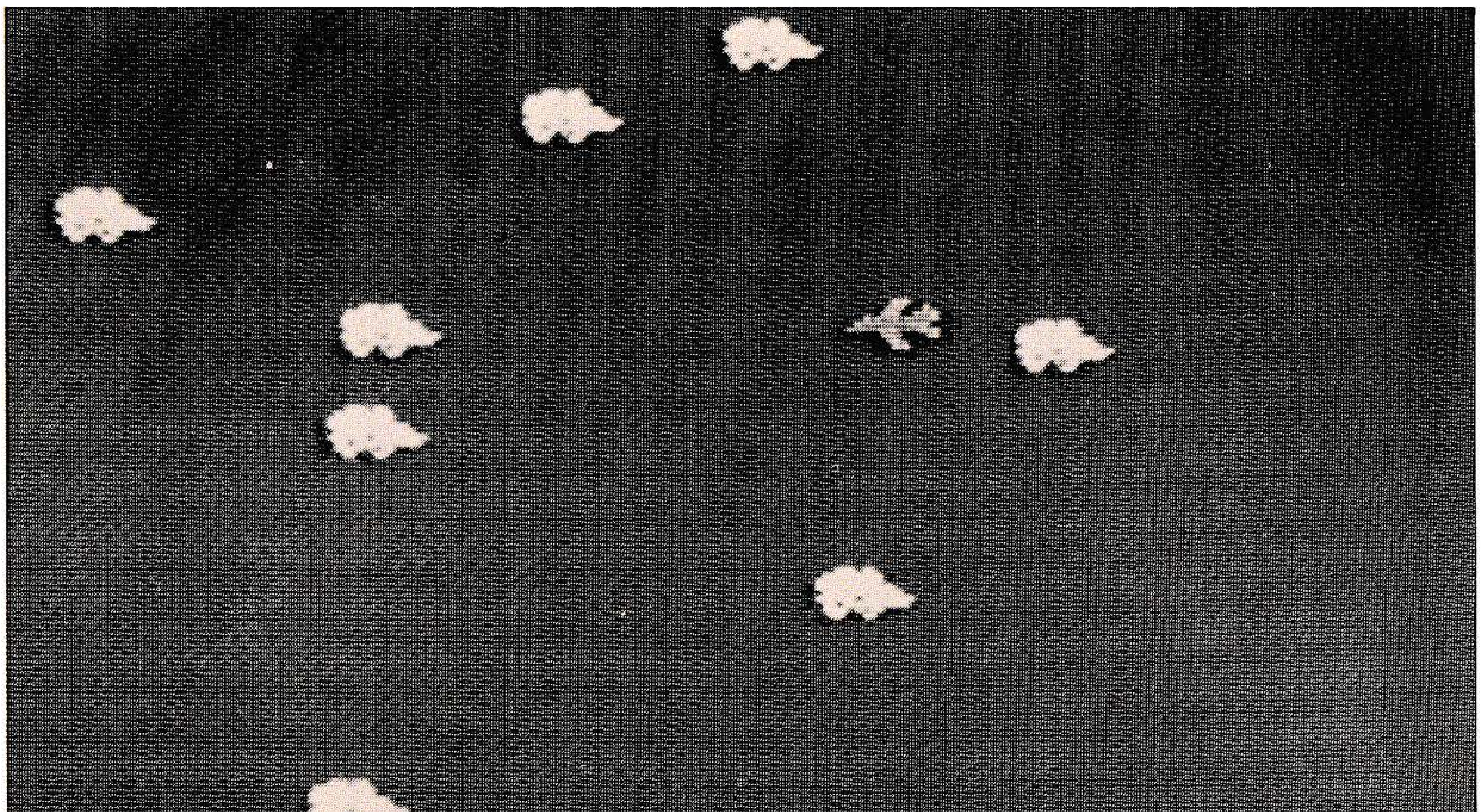
複雑な図形を素早く簡単に移動させるには，スプライト機能を使うと便利です．この機能を用いて，飛行機を飛ばしてみましょう．

飛行機は2色で表現したいので，スプライトパターンはそれぞれの色の部分について2つ作り，重ね合せて出力しています．この2つのスプライトパターンは，スプライト面の10番と11番に割り当てました．雲は，270〜300行で飛行機の割り当てられているスプライト面の前後に置き，飛行機が見え隠れするようにしています．

330〜420行は，カーソルキー入力に対する基本的なテクニックです．カーソルキー[↑]を押すと飛行機は上へ，[↓]を押すと下へ動いて行きます．330行では，スプライトが重なったときの割り込みを設定しています．そこで雲と飛行機が重なった場合は，440行のPLAY命令により雲との衝突音？を発します．

このプログラムは，ちょっと手を加えるだけで，ゲームになりそうですね．

なお，プログラムを終了するには，[←][→]キーを押してください．



List 9

```
100 ' List   9
110 '
120 DEFINT A-Z
130 DM=RND(-TIME)
140 XB=0       : YB=-1
150 XS=8       : YS=4
160 XE=255-16 : YE=191-16
170 '
180 SCREEN 2,2,0 : COLOR 15,4,4 : CLS
190 RESTORE 460
200 FOR I=0 TO 2
210   S$=""
220   FOR J=1 TO 32
230     READ R : S$=S$+CHR$(R)     スプライトパターンの設定
240   NEXT J
250   SPRITE$(I)=S$
260 NEXT I
270 FOR I=1 TO 20 STEP 2
280   X=(XE-XB)*RND(1)+XB : Y=(YE-YB)*RND(1)+YB
290   IF I<>11 THEN PUT SPRITE I,(X,Y),15,2
300 NEXT I
310 X=XE : Y=(YB+YE)/2
320 '
330 ON SPRITE GOSUB 440
340 SPRITE ON                      キー入力によって飛行機を動かすルーチン
350 I=STICK(0)
360   X=X-XS : IF X<XB THEN X=XE
370   IF I=1 AND Y-YS>YB THEN Y=Y-YS
380   IF I=5 AND Y+YS<YE THEN Y=Y+YS
390   IF I=3 OR I=7 THEN END
400   PUT SPRITE 10,(X,Y),9,0 : PUT SPRITE 11,(X,Y),11,1
410   FOR T=1 TO 20 : NEXT T
420 GOTO 340
430 '
440 SPRITE OFF : PLAY "V12L64O3A" : RETURN
450 '雲のスプライトパターン・データ
460 DATA 0,0,0,0,0,0,0,0,31,63,124,0,0,0,0,0
470 DATA 0,0,0,0,0,0,12,28,252,224,112,0,0,0,0,0
480 '飛行機のスプライトパターン・データ
490 DATA 0,0,0,0,0,0,1,3,0,0,3,3,1,0,0,0
500 DATA 0,0,0,0,0,224,192,130,2,28,142,134,192,224,0,0
510 '雲のスプライトパターン・データ
520 DATA 0,0,14,63,127,127,63,126,55,61,24,0,0,0,0,0
530 DATA 0,0,192,224,224,248,252,255,254,96,192,0,0,0,0,0
```

タイマー&エネルギーメータ

迫りくる時間，刻々と減っていくエネルギー，いずれもゲームにスリルと迫力を与えてくれる大切な小道具です．

そこで，コックピットに簡単なタイマー，エネルギーメータを描いてみましょう．

コックピットは，*Figure 1* のようなものを考えました．まずは，エネルギーメータですが，スペースバーを押すとビームが発射されエネルギーが減っていくというものです．この表示には380行のようにLINE命令を用いています

つぎに，コンパスですが，カーソルキーの ⇦ と ⇨ にDRAW命令の角度指定を対応させて“▲”を回転させています．

このプログラムと“4.3 ビーム砲を打つ”“4.9 飛行機を飛ばす”を組み合わせて，簡単なゲームを作ってみるのもおもしろいでしょう．

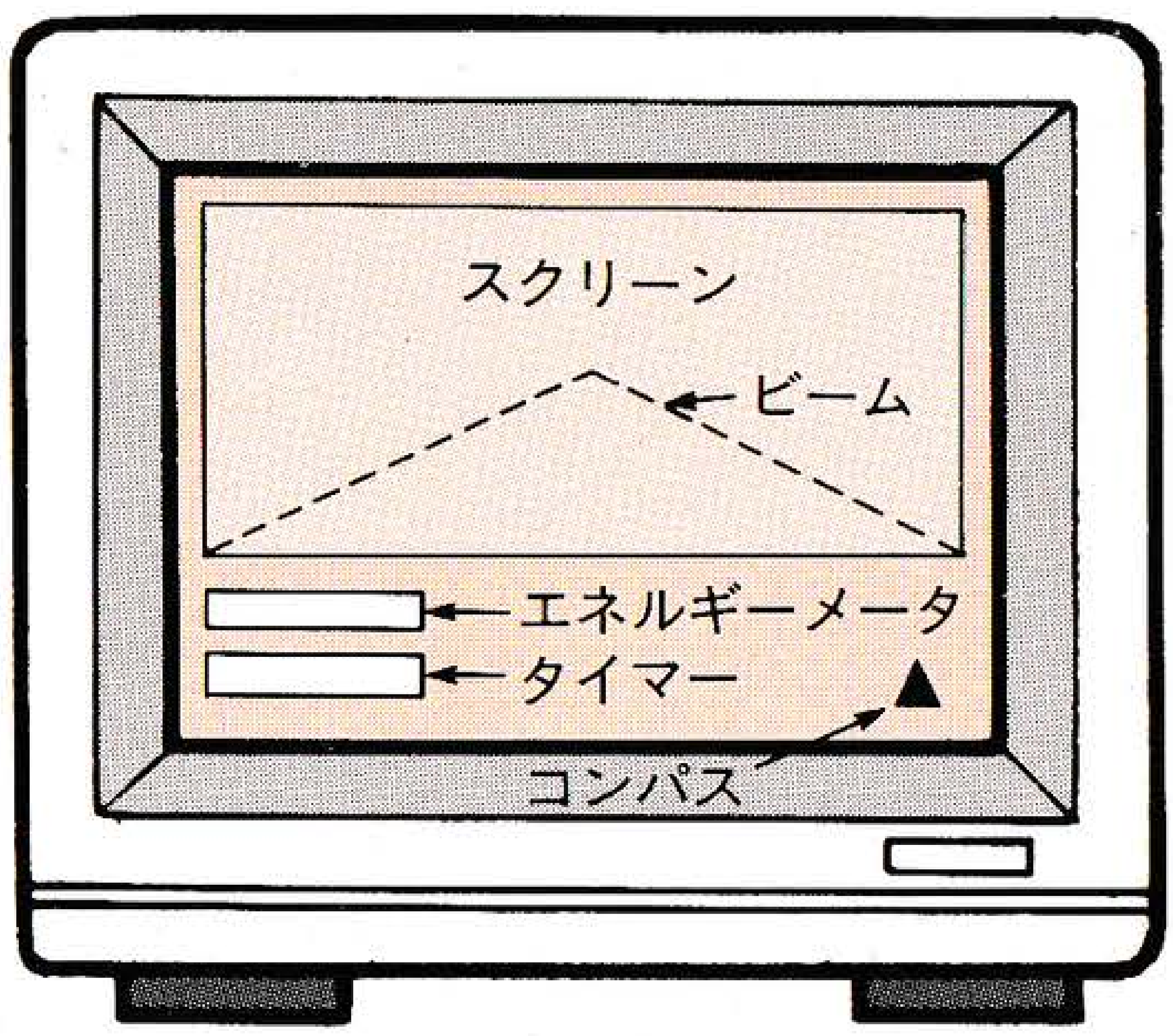
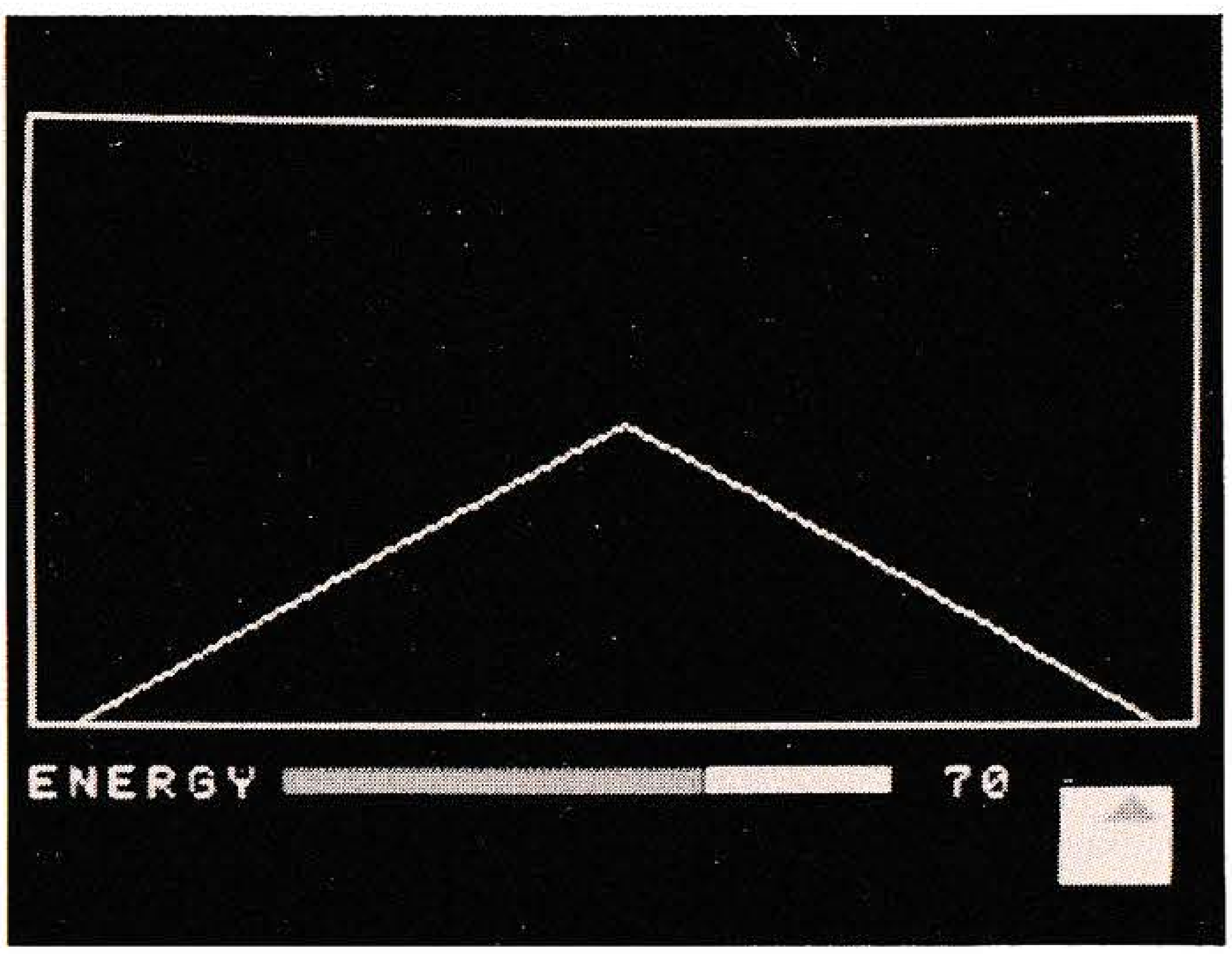
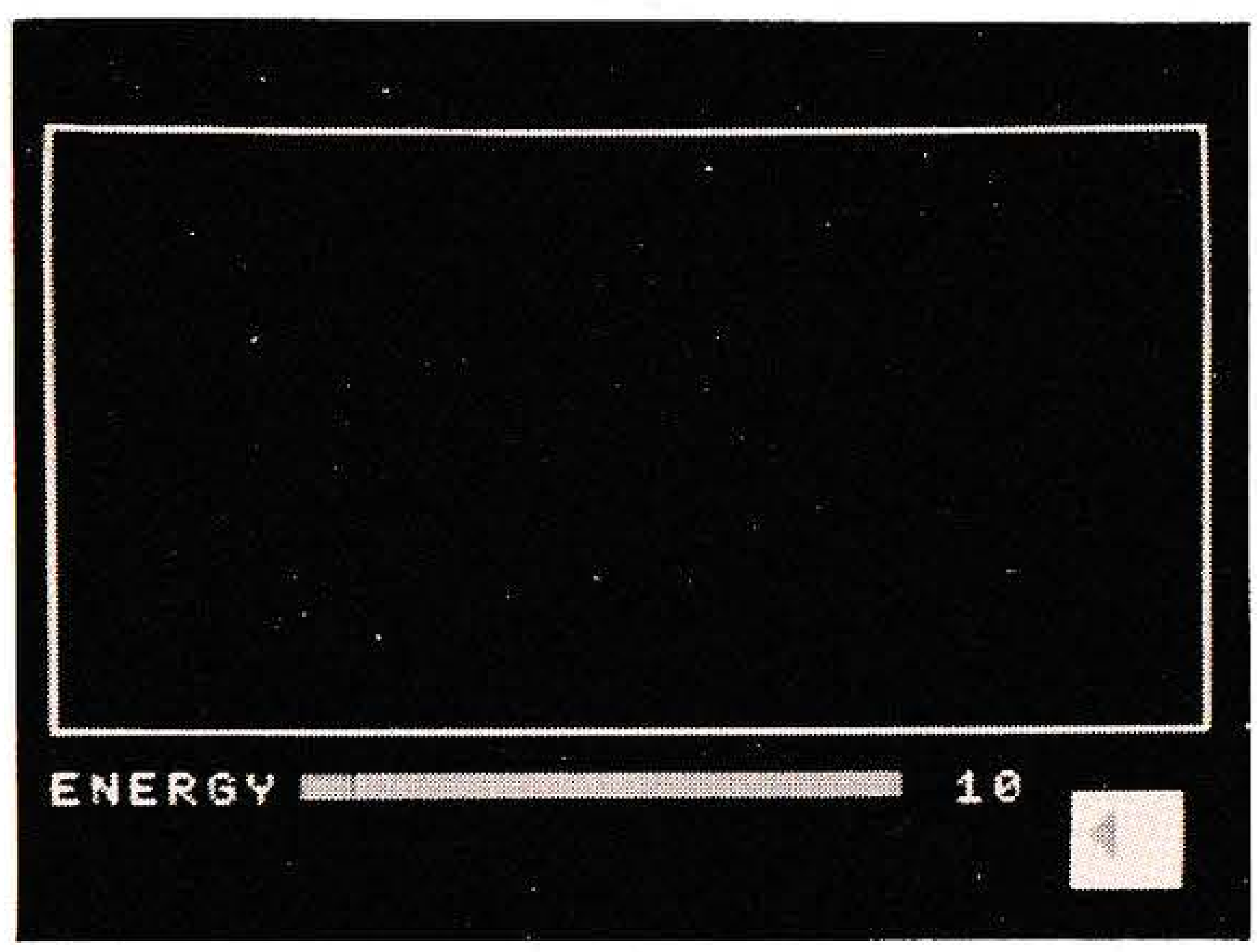


Figure 1





List 10

```
100 ' List   10
110 '
120 XS=60  : YS=160
130 XE=180 : YE=165
140 XA=225 : YA=172
150 A=0 : C=11 : CB=4 : EG=100
160 '
170 SCREEN 2 : COLOR 15,0,0 : CLS
180 OPEN "GRP:" FOR OUTPUT AS #1
190 LINE (10,10)-(245,150),15,B
200 LINE (XS,YS)-(XE,YE),CB,BF
210 DRAW "S4A0 BM10,=YS;" : PRINT #1,"ENERGY"
220 X=XE+4
230 DRAW "BM=X;,=YS;" : PRINT #1,"FULL"
240 NW=EG
250 GOSUB 470
260 '
270 I=STICK(0)
280   IF STRIG(0) THEN GOSUB 340
290   IF I=7 THEN A=A+1 : GOSUB 460
300   IF I=3 THEN A=A-1 : GOSUB 460
310 IF NW>0 THEN 270
320 PLAY "V12L1O2A" : DM$=INPUT$(2):END
330 '
340 NW=NW-10
350 X=XE-(XE-XS)*(EG-NW)/EG
360 LINE (X,YS)-(XE,YE),C,BF
370 X=XE+4 : DRAW "BM=X;,=YS;"
380 LINE (X,YS)-(X+30,YS+8),1,BF
390 DRAW "BM=X;,=YS;" : PRINT #1,USING "###";NW
400 IF NW<=EG/2 THEN C=8
410 LINE (20,149)-(128,80),15 : LINE (235,149)-(128,80),15
420 PLAY "T128L64O5A" : FOR T=1 TO 80 : NEXT T
430 LINE (20,149)-(128,80),1  : LINE (235,149)-(128,80),1
440 RETURN
450 '
460 IF A<0 THEN A=3 ELSE IF A>3 THEN A=0
470 LINE (XA-9,YA-7)-(XA+14,YA+14),15,BF
480 XB=(A¥2)*8 : YB=(((A+1) MOD 4)¥2)*8
490 X=XA+XB : Y=YA+YB
500 DRAW "BM=X;,=Y;A=A;C8R12H6G6BR6BU2"
510 PAINT STEP(0,0),8
520 RETURN
```

メインルーチン (270–310)

エネルギーメータを描くサブルーチン (340–440)

タイマーを描くサブルーチン (460–520)

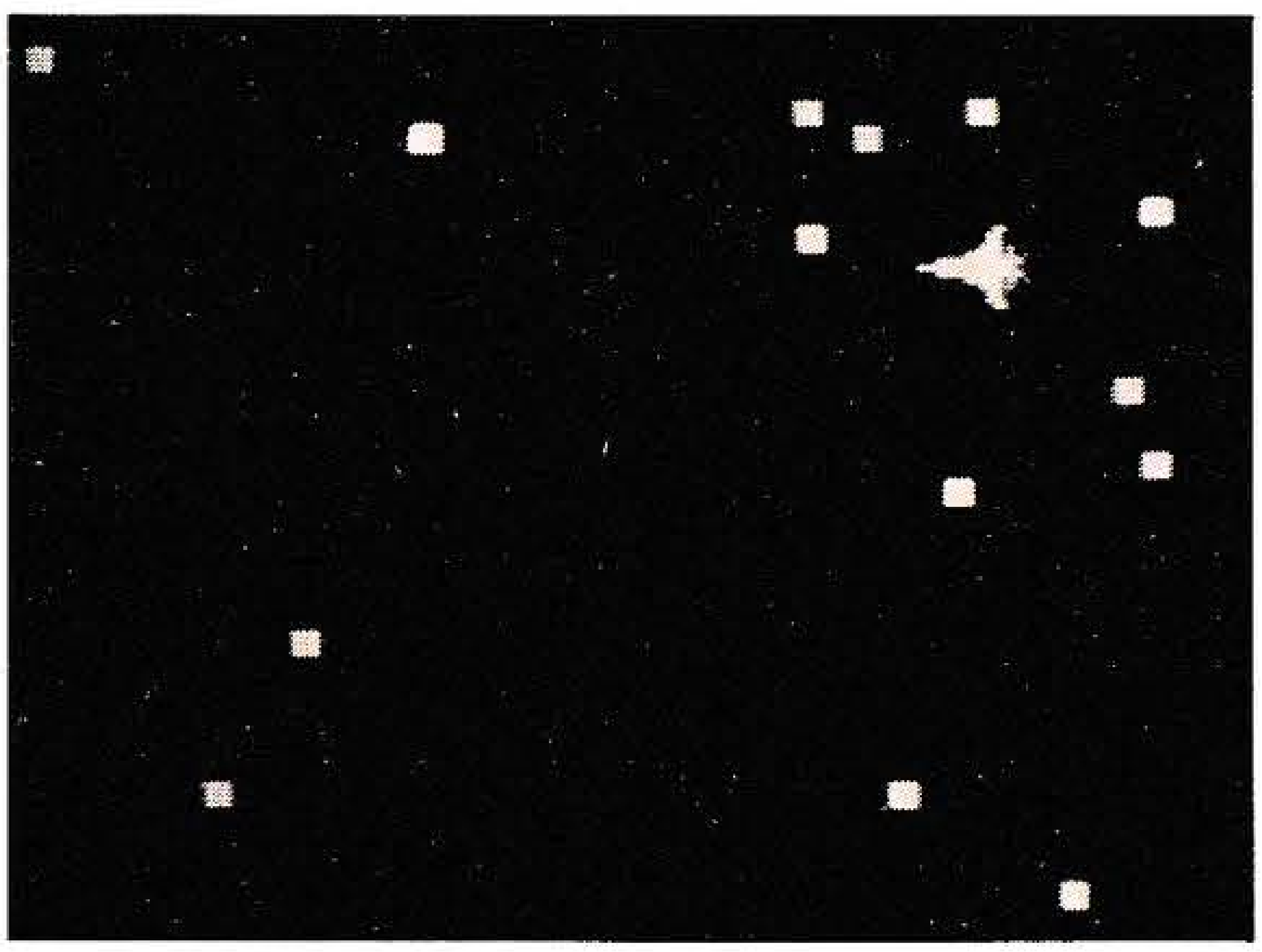
流れる星空

画面の下地が次々と下方や左右に流れていく中で場面が展開していくタイプのゲームを，スクロールゲームと呼びます．狭いディスプレイ画面に閉じ込もることなく，背景に広大な世界を構築できるスクロールゲームは，ゲームセンターでも常に高い人気を得ています．

ここでは，スクロールの最も基本的なパターンとして，星座のように一定の形で並んだ星が左から右へと流れていくプログラムを作ってみましょう．そしてその星空を背景に，一隻の宇宙船が東の空へと飛んでいきます．

星を動かしている手順を順番に考えてみると次のようになります．

(1) 星を消す
(2) 星の位置を DS 分だけ右へ進め，その座標が右端の座標より大きくなったらそれを左端へもどす．



(3) 星を描きなおす．
(4) (1) の処理にもどる．

星を同時にたくさん動かしたいときは，この処理を1つひとつの星について行えばよいのですが，仮りにそれを1つひとつプログラムに書いたら，それは長大なものになってしまいます．そこで，この場合のように，同じ処理を何度も繰り返し行わせるときには， FOR～NEXT ループと配列変数を組み合わせると便利です．

実際，この処理とプログラムの対応を見てみると，320～340行で (1)，(2)，(3) の操作を行い，I 番めの星の移動を行います．そして FOR～NEXT ループで変数 I の値を1から DN まで変化させながら同様の処理を行い，すべての星の位置を変えているわけです．

このように，多くの点や図形を，同時に同じパターンで移動させる場合には，配列変数とループを組み合わせることが一番簡単な方法です．

宇宙船を表示する部分は380～440行にあたります．ここでは STICK (0) を使ったカーソルキー入力により，その位置を動かしています．宇宙船はスプライト機能で表示するため，背景の星空に影響なく上下に操作できます．

なお，スペースバーを押すと，ビーム砲が発射されます．

List 11

```
100 ' List   11
110 '
120 DN=30  : DS=24
130 JX=180 : JY=90 : JC=13 : JS=16
140 DIM DX(DN),DY(DN),DC(DN)
150 JT$="" : RESTORE 520
160 FOR I=1 TO 32
170   READ D : JT$=JT$+CHR$(D)
180 NEXT I
190 '
200 SCREEN 3,2 : COLOR 15,1,1 : CLS
210 SPRITE$(0)=JT$
220 FOR I=1 TO DN
230   DX(I)=INT(RND(1)*256)
240   DY(I)=INT(RND(1)*192)
250   DC(I)=INT(RND(1)*14)+2
260   PSET (DX(I),DY(I)),DC(I)
270 NEXT I
280 '
290 GOSUB 310 : GOSUB 380 : GOSUB 460 : GOTO 290
300 '
310 FOR I=1 TO DN
320   DX=DX(I) : DY=DY(I)
330   PRESET(DX,DY) : DX=(DX+DS) MOD 256
340   PSET (DX,DY),DC(I) : DX(I)=DX
350 NEXT I
360 RETURN
370 '
380 R=STICK(0)
390   IF R=1 THEN J=-JS
400   IF R=5 THEN J=JS
410   JN=JY+J : PLAY "V12T255L64O6A"
420   IF JN<0 OR JN>175 THEN RETURN
430   JY=JN : PUT SPRITE 0,(JX,JY),JC,0
440 RETURN
450 '
460 IF NOT STRIG(0) THEN RETURN
470   LINE (0,JY+8)-(JX,JY+8),8
480   PLAY "T255L64N60N61N62N63N64"
490   LINE (0,JY+8)-(JX,JY+8),1
500 RETURN
510 '
520 DATA 0,0,0,0,0,1,31,255,31,1,0,0,0,0,0,0
530 DATA 0,24,48,48,116,245,254,252,254,245,116,48,48,24,0,0
```

150–180: スプライトパターンの定義

220–270: 星を描く

310–360: 星を動かすサブルーチン

380–440: キーにより宇宙船を動かすサブルーチン

460–500: ビーム砲を描くサブルーチン

520–530: 宇宙船のスプライトパターン・データ

カニさんカニさん

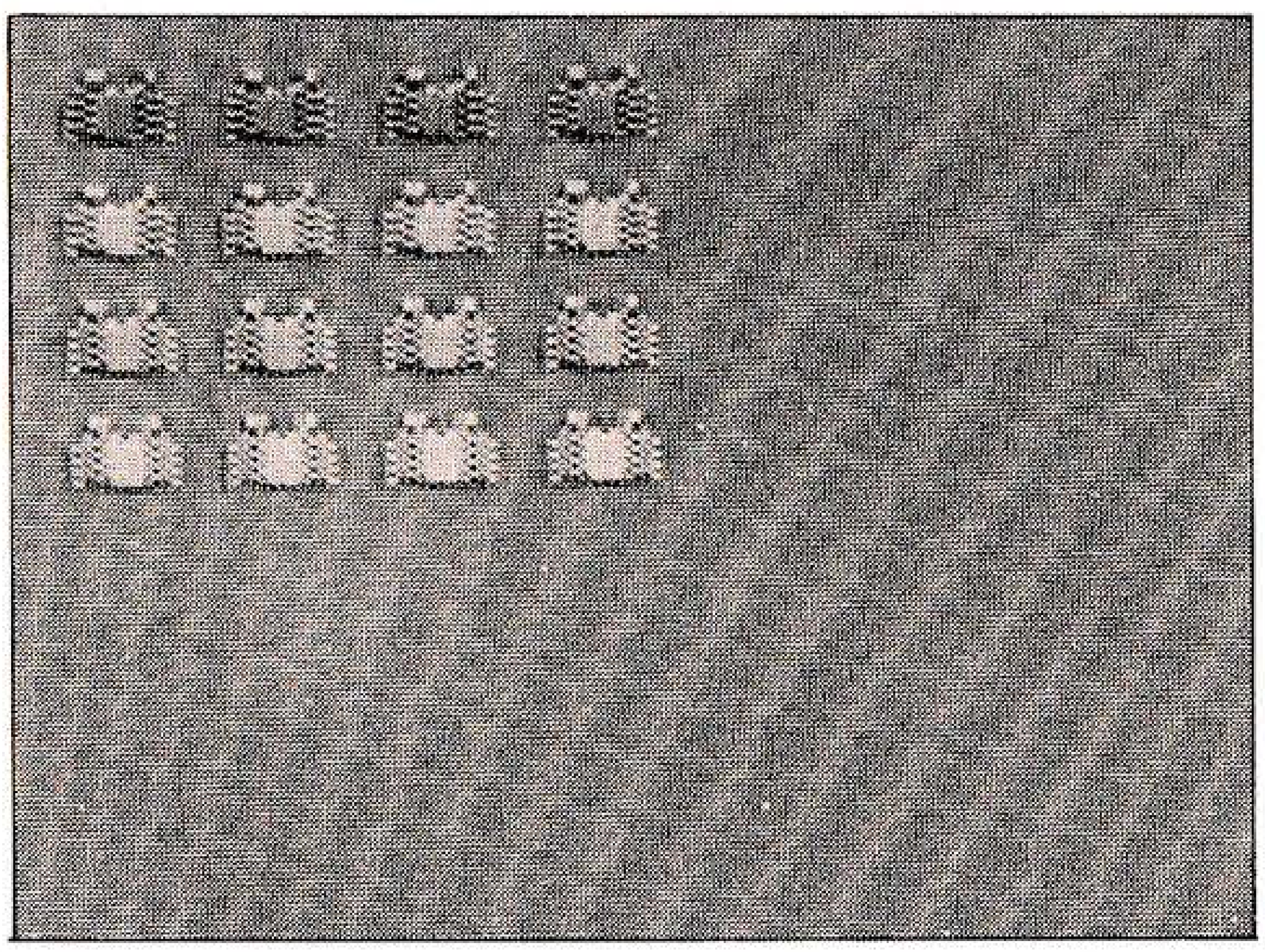
“流れる星空”のプログラムでは，同時に多くの図形を動かすテクニックの基本について述べました．ここではそれを応用し，カニが16ぴきいっせいにムッショ，ムッショ，ムッショと動いていくプログラムを作り，その仕組みを見ていくことにしましょう．

まず *List 12a* を実行してみると，1ぴきのカニが足を上下に振りながら，画面を左右へ往復し，一段ずつ下へ降りてきます．下まで達したところでプログラムは終了となります．

このカニは，スプライト機能によって表示されています．このスプライトパターンの定義は，170～250行で行われています．

このようにAとBの2種類，カニのパターンを用意し，それを交互に出力してカニの足に動きを与えるわけです．

カニは，右端と左端で一段下に降りますが，この判断は，340行で行っています．*Figure 1* を見てください．左端にカニが達したかどうかは，Xの値と左端の座標をくらべればOKです．これに対してカニが右端に達したかどうかは，X＋15の値で判断しなければなりません．もし，両端のどちらかに達した場合には，350行でカニのy



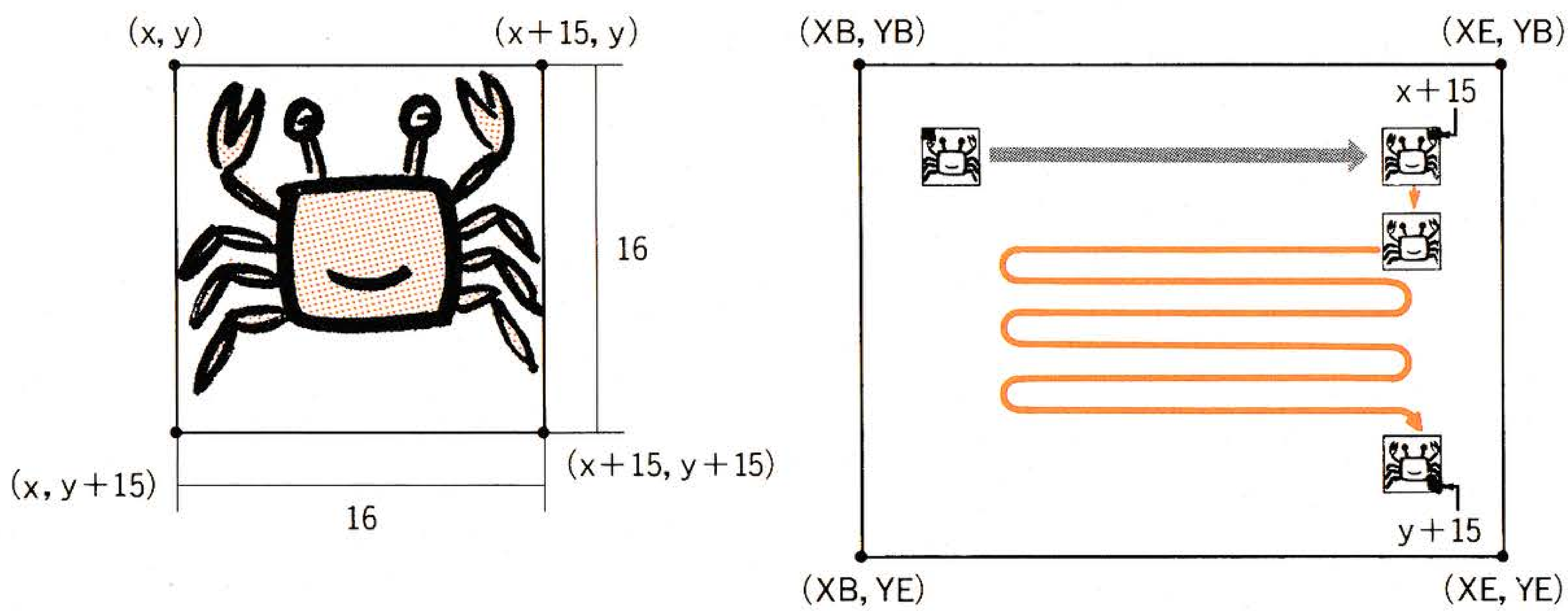


Figure 1

座標（縦方向）はYSだけ下にずらされます．さらに，その新しい値Yと画面の下端の座標YEをくらべ，プログラムの終わりを判断しています．この判断は，360行で行われていますが，ここでもくらべられている値はY+15とYEであるということに注意してください．

さて，以上のプログラムを理解することができたでしょうか．1ぴきのカニを動かせるようになれば，これを16ぴきにふやすことは簡単です．その例としてのプログラムを *List 12b* にあげました．2つのプログラムを見くらべて，いったいどこが変わっているのか，考えてみてください

List 12a

```
100 ' List  12a
110 '
120 XB=10  : YB=10
130 XE=250 : YE=180
140 XS=4   : YS=20
150 X=XB   : Y=YB
160 '
170 RESTORE 390 : A$=""
180 FOR I=1 TO 32
190   READ A : A$=A$+CHR$(A)
200 NEXT I
210 '
220 RESTORE 420 : B$=""
230 FOR I=1 TO 32
240   READ B : B$=B$+CHR$(B)
250 NEXT I
260 '
270 SCREEN 2,2 : COLOR 15,4,4 : CLS
280 SPRITE$(0)=A$ : SPRITE$(1)=B$ : F=0
290 '
300 F=1-F : PLAY "V12T128L32O2A"
310 PUT SPRITE 0,(X,Y),8,F
320 X=X+XS
330 FOR T=1 TO 100 : NEXT T
340 IF X>=XB AND X+15<=XE THEN 300
350 Y=Y+YS : XS=-XS
360 IF Y+15<YE THEN 300
370 END
380 '
390 DATA 0,0,0,20,20,8,42,87,175,87,175,87,175,87,139,0
400 DATA 0,0,0,40,40,16,84,234,245,234,245,234,245,234,209,0
410 '
420 DATA 0,32,80,48,10,7,63,71,191,71,191,71,187,64,128,0
430 DATA 0,4,10,12,80,224,252,226,253,226,253,226,221,2,1,0
```

（170〜250行：スプライトパターンの定義）

（280〜360行：メインルーチン）

List 12b

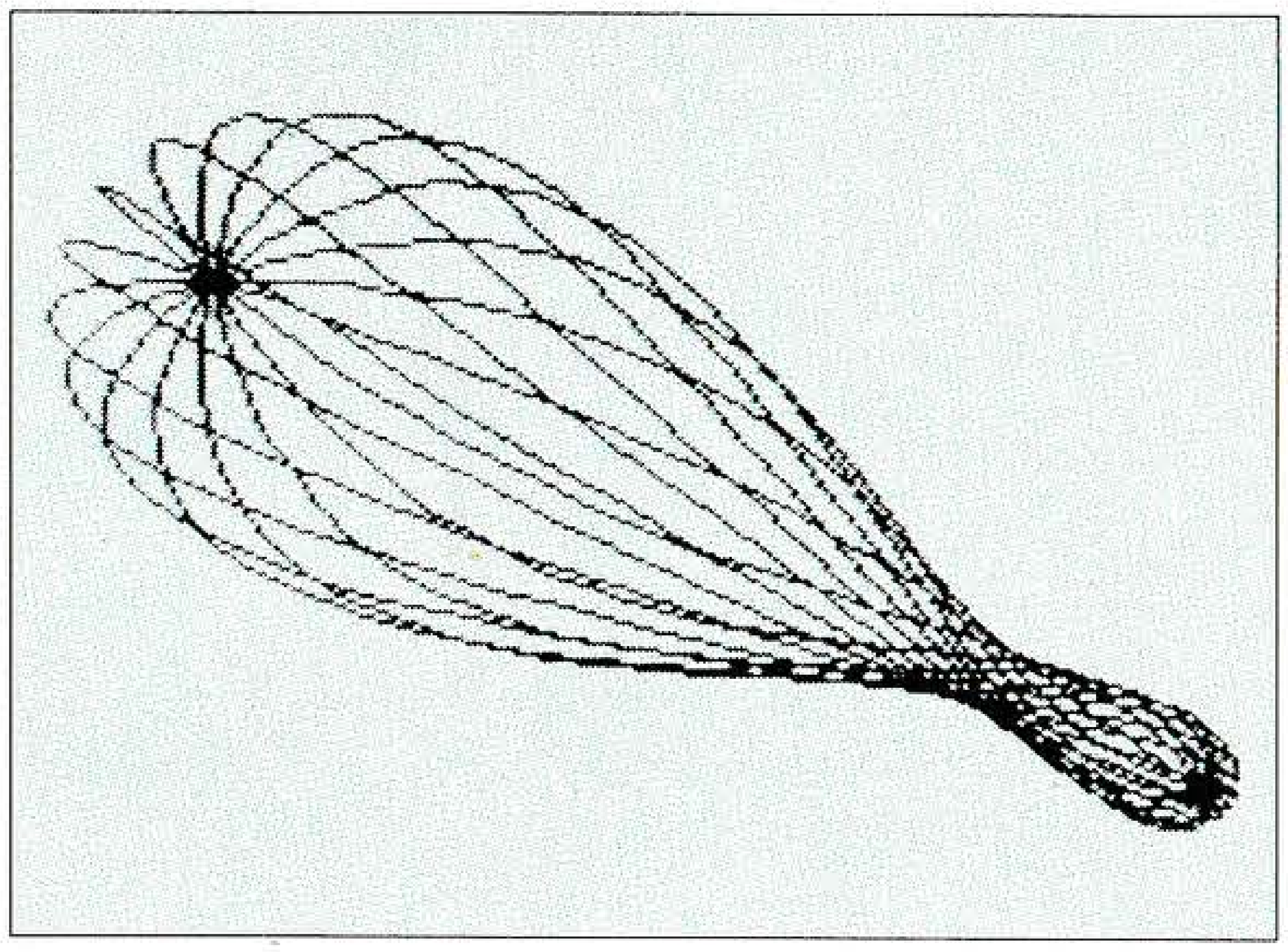
```
100 ' List   12b
110 '
120 IN=4    : JN=4
130 XB=10   : YB=10
140 XE=250  : YE=180
150 XS=4    : YS=20
160 N=IN*JN-1 : DIM X(N),Y(N),C(N)
170 '
180 RESTORE 500 : A$=""
190 FOR I=1 TO 32
200   READ A : A$=A$+CHR$(A)
210 NEXT I
220 '
230 RESTORE 530 : B$=""
240 FOR I=1 TO 32
250   READ B : B$=B$+CHR$(B)
260 NEXT I
270 '
280 SCREEN 2,2 : COLOR 15,4,4 : CLS
290 SPRITE$(0)=A$ : SPRITE$(1)=B$ : F=0
300 FOR I=0 TO N
310   X(I)=XB+(I MOD IN)*24
320   Y(I)=YB+(I ¥ IN)*20
330   C(I)=(I ¥ IN)+8
340 NEXT I
350 '
360 F=1-F : PLAY "V12T128L32O2A"
370 FOR I=0 TO N
380   PUT SPRITE I,(X(I),Y(I)),C(I),F
390   X(I)=X(I)+XS
400 NEXT I
410 FOR T=1 TO 100 : NEXT T
420 IF X(0)>=XB AND X(N)+15<=XE THEN 360
430 FOR I=0 TO N
440   Y(I)=Y(I)+YS
450 NEXT I
460 XS=-XS
470 IF Y(N)+15<YE THEN 360
480 END
490 '
500 DATA 0,0,0,20,20,8,42,87,175,87,175,87,175,87,139,0
510 DATA 0,0,0,40,40,16,84,234,245,234,245,234,245,234,209,0
520 '
530 DATA 0,32,80,48,10,7,63,71,191,71,191,71,187,64,128,0
540 DATA 0,4,10,12,80,224,252,226,253,226,253,226,221,2,1,0
```

スプライトパターンの定義 (180–260)

メインルーチン (280–480)

いちじく関数？

座標（X0, Y0）から速度（U0, V0）で出発し，座標（X1, Y1）へ速度（U1, V1）で到着する物体が，その２点間でたどる道筋を考えてみます．x座標，y座標ともに媒介変数tの3次関数（０≦ｔ≦１）として変化すると仮定すれば，簡単な計算で，下記のプログラムの280～300行に示したような式を得ることができます．ただしこの中のFNA(T)～FND(T)は120～150行で定義している3次関数です（これらの式がどのようにして導き出されたものか私は知っているが，ここにそれを書き記すだけの余地はない．詳細は，フェルマー氏に聞いてほしい！）．

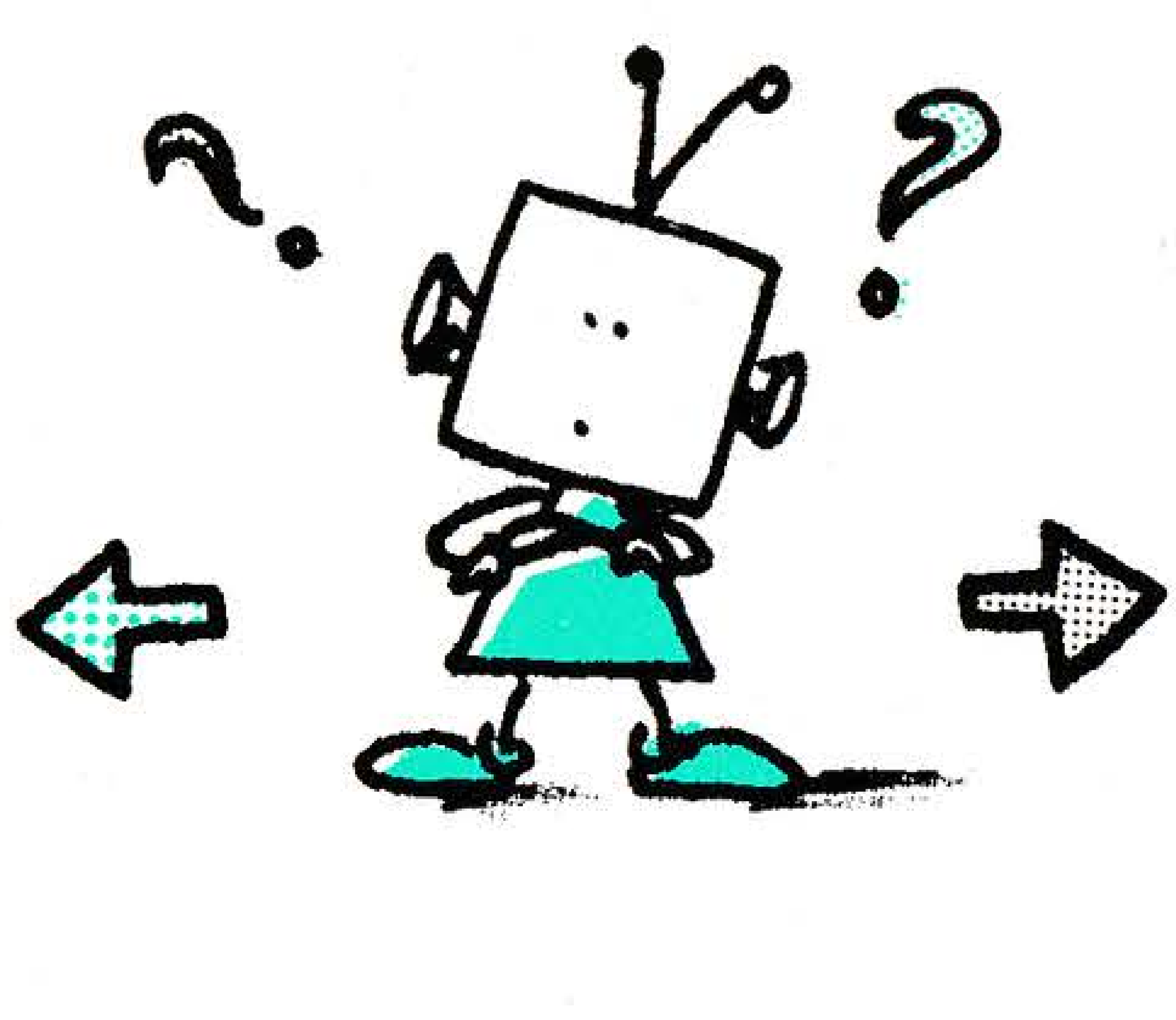


```
100 ' List Ex.4
110 '
120 DEF FNA(T)=(T-1)*(T-1)*(T+T+1)
130 DEF FNB(T)=T*T*(3-T-T)
140 DEF FNC(T)=(T-1)*(T-1)*T
150 DEF FND(T)=(T-1)*T*T
160 '
170 SCREEN 2 : COLOR 1,3,3 : CLS
180 DG=3.14159/180 : R0=360 : R1=160
190 FOR I=0 TO 359 STEP 20
200   T0=I*DG : T1=(I+15)*DG
210   X0=48 : X1=208 : U0=R0*COS(T0) : U1=R1*COS(T1)
220   Y0=48 : Y1=144 : V0=R0*SIN(T0) : V1=R1*SIN(T1)
230   GOSUB 270
240 NEXT I
250 I$=INPUT$(1):END
260 '
270 FOR T=0 TO 1 STEP 1/32
280   A=FNA(T) : B=FNB(T) : C=FNC(T) : D=FND(T)
290   X=A*X0+B*X1+C*U0+D*U1
300   Y=A*Y0+B*Y1+C*V0+D*V1
310   IF T=0 THEN PSET (X,Y),1
320   IF T>0 THEN LINE-(X,Y),1
330 NEXT T
340 RETURN
```

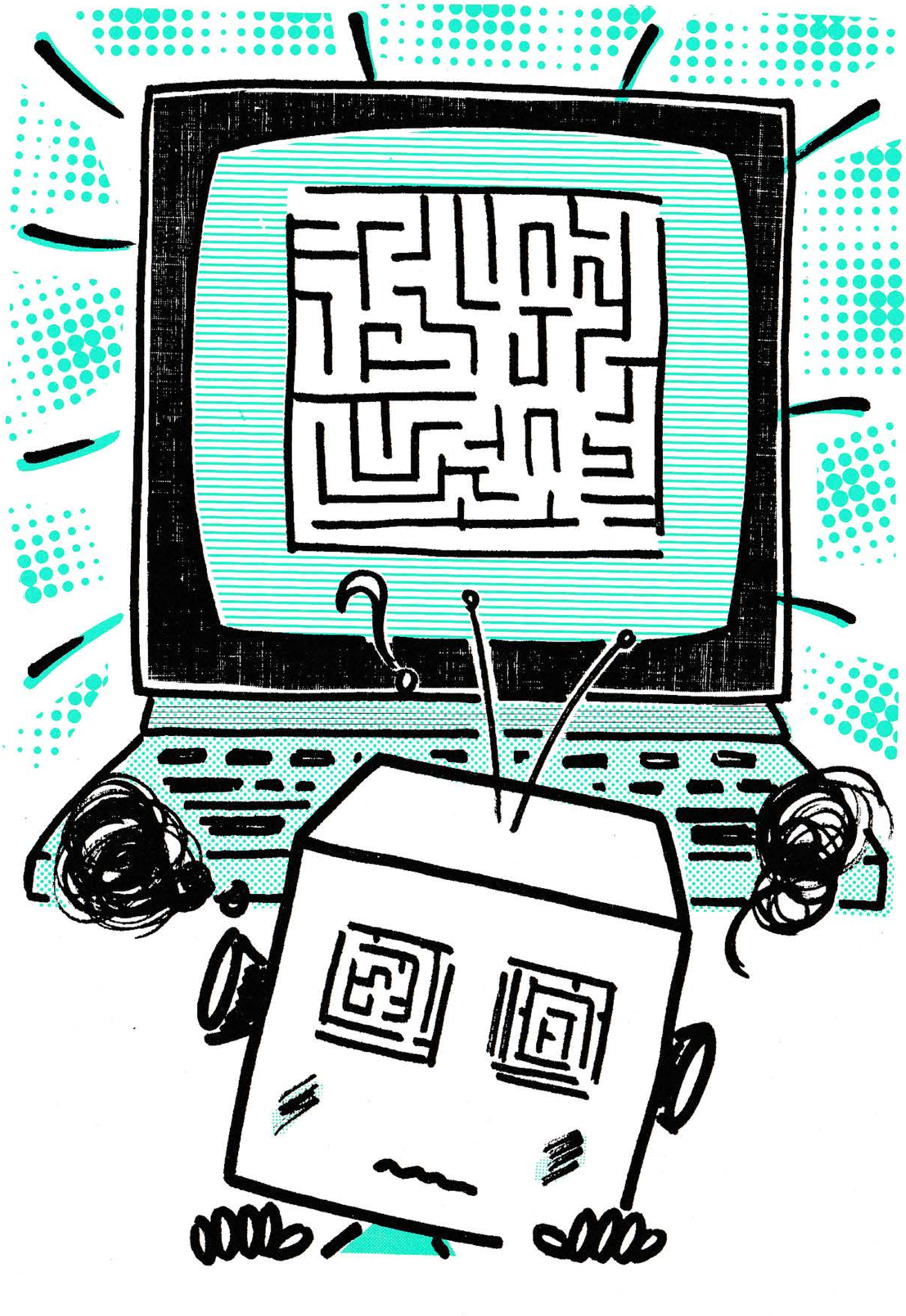
【5章】

ランダムアプリケーション

バイオリズム
迷路ゲーム
焼けてないトタン屋根の上のネコ
スキーゲーム



本書をここまで読んできた方は，コンピュータゲームやコンピュータ・グラフィックスを見たときに，およそどんなプログラムになっているか見当がつくはずです．しかし，いざ自分でプログラムを作ろうと思うと，今までの1つひとつのブロックを，どうやって組み合わせていけばよいのか，どうもよくわからないという人もいることと思います．そこで，あくまでもサンプルとしてですが，簡単なゲームを実際に作ってみることにしました．しかし，プログラムは自分で作ってこそ上達するものです．これら既存のゲームで遊ぶだけではなく，自分でも，プログラムを作ってみてください．また，このプログラムを自分なりに変更してみるのも，プログラミング上達には有効な方法でしょう．

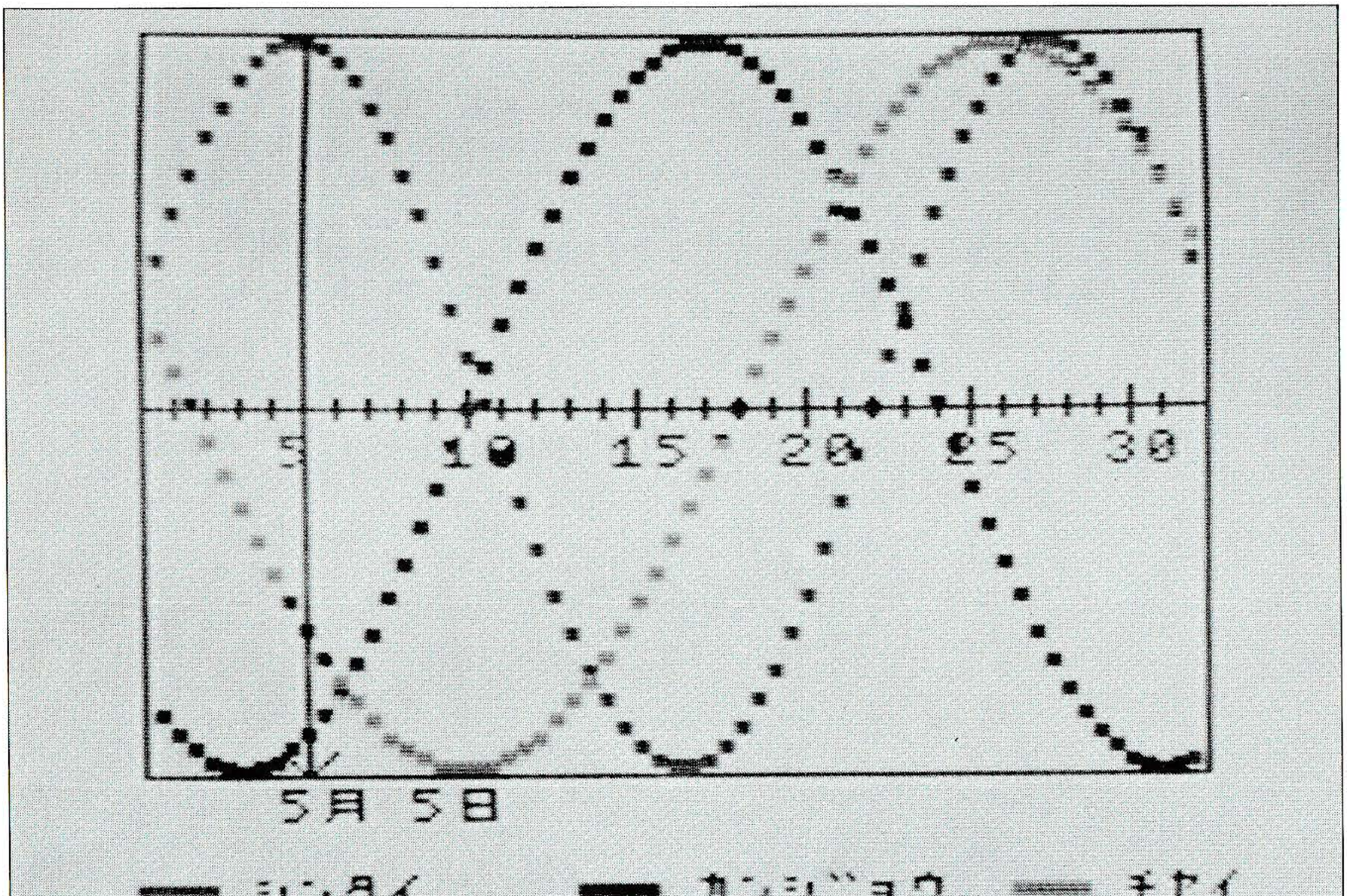


バイオリズム

本書の中にも何度か登場した，sin，cos関数のカーブ，あれを見て，みなさんは何を連想するでしょうか?　海の波，それともヘビか何かが動いている様子でしょうか?実は，ゲームセンターなどでよく見かけるバイオリズムのあのカーブも，sin 関数を使って描かれているものなのです．では，さっそくそのバイオリズムのカーブを出力してみることにしましょう．

利用法

プログラムを実行すると，まず名前を聞いてきますので，入力してください．次に，あなたの誕生日の入力をうながしてきますので，入力してください．このとき，年はかならず西暦で入力します．つまり昭和59年ならば1984年となるのです．最後に，予想日を入力します．



ただし，誕生日から予想日まで，89年以上離れている場合，たとえば1984年現在満89歳以上の人では，その生きていた日々の計算をしきれずに，“Over flow error”を出力します(ごめんなさい！)．たとえばイエス・キリストのバイオリズムを見たいなどという恐れ多いことはお控えください．

なお，存在しない日付や月を入力すると，再度入力をうながしてきます．

その後，身体（P），感情（S），知性（I）についての，それぞれの曲線が，予想日を含めた1か月分出力されます．横軸の目盛りは，1日の長さを示しています．さらに，予想日は，白い矢印で示されます．この矢印が引かれる位置は，その予想日が始まる真夜中の0時になっていることに注意してください．

プログラムの説明

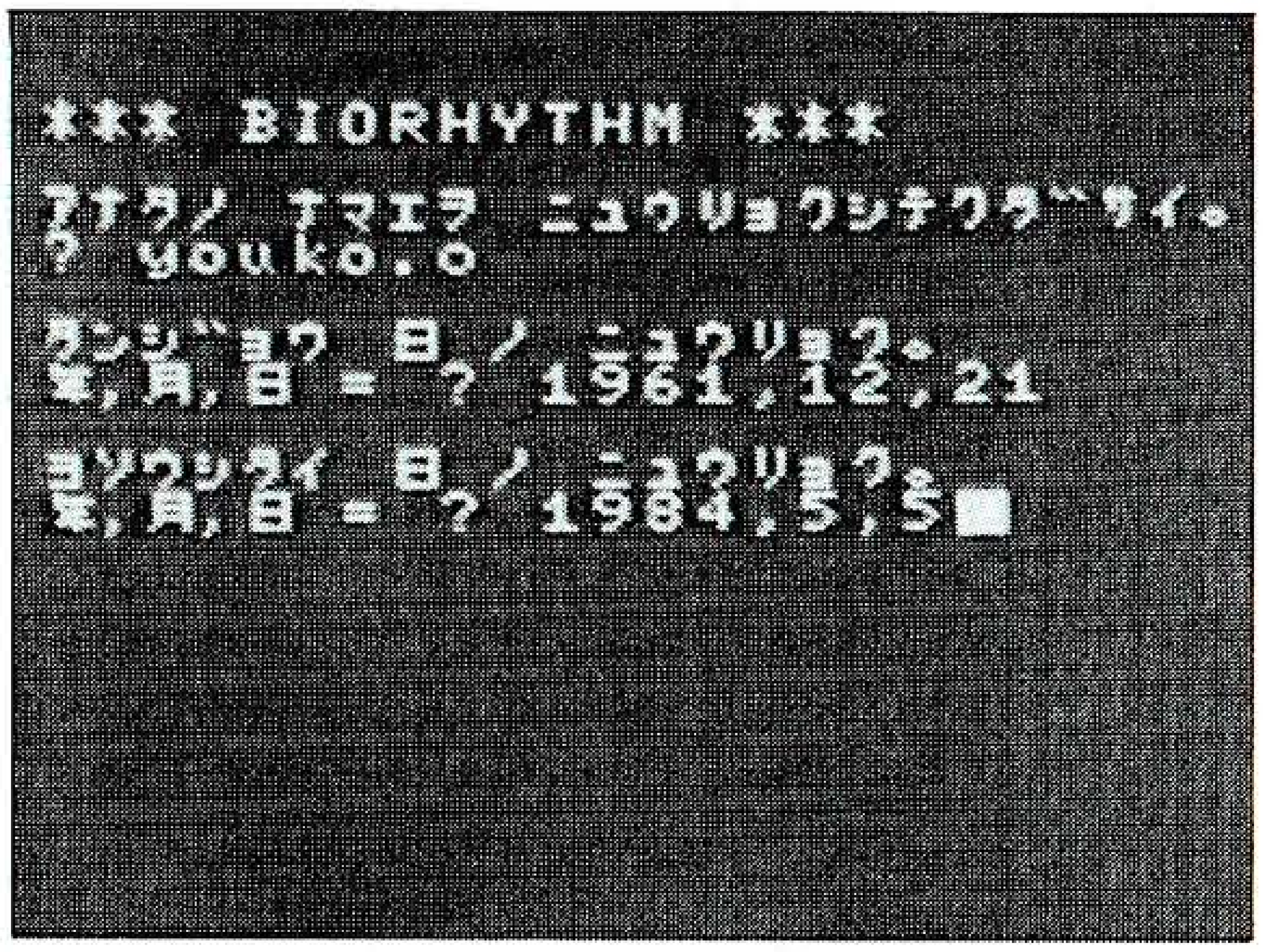
このバイオリズムのプログラムは，大きく分けて2つの部分から成ります．1つは，今まで生きてきた日数を計算する部分（このとき，うるう年に注意！），もう1つは計算したデータをもとに，サインカーブを描かせる部分（身体，感情，知性で，それぞれ，周期が異なる）です．

● 190〜230行

名前，誕生日，予想日の入力部分です．日付の入力には，310〜480行のサブルーチンをコールします．

● 250行

コメントを書くため，980行以降のサブルーチンをコールします．



● 260行

グラフィックの表示を行うため，520行以降のサブルーチンをコールします．

● 310〜480行

この部分は，日付をインプットし，さらに西暦0年1月1日から，予想日までの日数を計算するサブルーチンです．

まず，1月〜12月までの月の長さを，それぞれ配列変数ML(n)に読み込みます．次に，370行で年，月，日を入力したあと，380，390行でその年がうるう年かどうかを判断しています．うるう年は基本的には4で割り切れる年ですが，100で割れて400で割れない年はうるう年ではありません．380行の式は，うるう年ならば，F=1，そうでなければF=0という結果を与えます．したがって，うるう年ならば，390行でML（2）が29日に変更されます．400，410行で，M月D日という日付が実際に存在するかどうかチェックします．

430行では，指定したY年の前年までのすべての日数を計算しています．さらに，440〜460行で，M月の前の月までの日数を足し，最後に470行でDを足して合計の日

数を求めています．

● 520～940行

ここでは，それぞれのグラフを出力します．まず520行で，身体 PW，感情 SW，知性 IW の周期を決めます．530行では，誕生日から予想日の月はじめまでの日数を計算し，変数 DD に代入しています．この DD は，あなたが生まれてから，予想日の月はじめまで生きてきた日数です．

540行は，身体，感情，知性の周期で DD を割り，その余りを求めることで，予想日の月はじめが，それぞれの周期のどこに位置しているかを判定しています．

570行はグラフの枠を描いている部分です．

590～630行では，身体，感情，知性について，それぞれの周期をもとに，グラフを描いています．さらに，月はじめにそれぞれの周期のどの位置にあるのかを計算した結果である P，S，I と，日付を進めていく変数 T を足しています．

● 660～690行は，グラフに日付の目盛りを入れます．

● 700～760行でこの目盛りの下に日付を表示します．

● 780～820行では，指定した予想日を矢印で示し，その日付を表示しています．

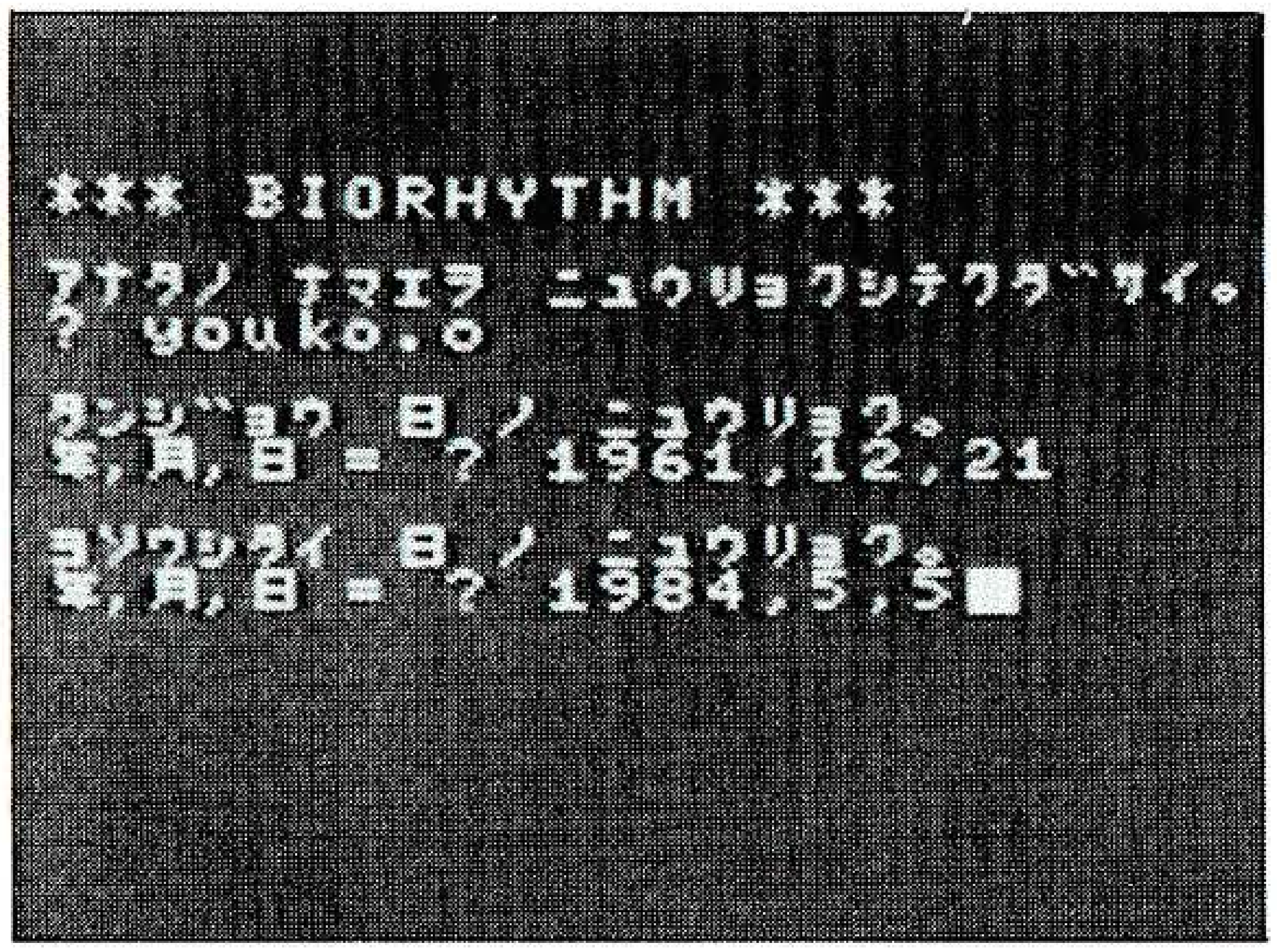
● 840～910行では，グラフの説明を表示しています．

● 930，940行は，グラフを描くときに点をプロットするためのサブルーチンです．

● 980～1100行

これは，日付を表示し，画面の右にバイオリズムの簡単な説明を出力している部分です．

バイオリズムの読み取り方としては「カーブが真ん中の軸と交差する日を要注意とし，事故などを起こしやすいので気をつける」というのが一般的です．なお，詳しくは，巻末の参考文献をご覧ください．



List 1

```
100 ' List   1
110 '
120 DEFSNG A-Z
130 DIM ML(12)
140 CP=5 : CS=8 : CI=11
150 P2=3.14159*2
160 SCREEN 1 : WIDTH 29 : COLOR 15,4,4 : CLS
170 KEY OFF
```

```
180 '
190 LOCATE 0,3 : PRINT "*** BIORHYTHM ***" : PRINT
200 PLAY "V10T250L8O3AR4AR4"
210 PRINT "ｱﾅﾀﾉ ﾅﾏｴｦ ﾆｭｳﾘｮｸｼﾃｸﾀﾞｻｲ｡" : INPUT NA$ : PRINT
220 MS$="ﾀﾝｼﾞｮｳ 日 ﾉ ﾆｭｳﾘｮｸ｡": GOSUB 310: MB=M: DB=D: YB=Y: BL=L
230 MS$="ﾖｿｳｼﾀｲ 日 ﾉ ﾆｭｳﾘｮｸ｡": GOSUB 310: MF=M: DF=D: YF=Y: FL=L
240 '
250 GOSUB 980
260 GOSUB 520
270 DM$=INPUT$(1) : COLOR 15,4,4 : END
280 '
290 ' 西暦0年1月1日から,INPUTされた年月日までの日数を計算するルーチン
300 '
310 RESTORE 350
320 FOR T=0 TO 12
330   READ ML(T)
340 NEXT T
350 DATA 0,31,28,31,30,31,30,31,31,30,31,30,31
360 '
370 PRINT MS$ : INPUT "年,月,日 = ";Y,M,D : PRINT
380 F=-((Y MOD 4)=0)+((Y MOD 100)=0)-((Y MOD 400)=0)
390 ML(2)=28+F
400 IF M<1 OR M>12 THEN PRINT M;"月 ? ﾅﾝﾉｺﾄﾃﾞｽｶ ?" : PRINT : GOT
O 370
410 IF D<1 OR D>ML(M) THEN PRINT M;"月";D;"日 ﾊ､ ｱﾘﾏｾﾝ !" : PRINT
: GOTO 370
420 '
430 YL=Y-1 : L=YL*365+INT(YL/4)-INT(YL/100)+INT(YL/400)
440 FOR T=0 TO M-1
450   L=L+ML(T)
460 NEXT T
470 L=L+D
480 RETURN
490 '
500 ' バイオリズム曲線を描くサブルーチン
510 '
520 PW=23 : SW=28 : IW=33
530 FL=FL-DF : DD=FL-BL
540 P=DD MOD PW : S=DD MOD SW : I=DD MOD IW
550 '
560 SCREEN 2 : COLOR ,15,15 : CLS
570 LINE (30,0)-(225,157),4,B
580 '
590 FOR T=.5 TO 32 STEP 1/2: X=T*6+30
600   Y=79-77*SIN((T+P)/PW*P2) : C=CP : GOSUB 930
610   Y=79-77*SIN((T+S)/SW*P2) : C=CS : GOSUB 930
620   Y=79-77*SIN((T+I)/IW*P2) : C=CI : GOSUB 930
630 NEXT T
```

```
640 '
650 LINE (30,79)-(225,79),4
660 FOR T=1 TO 31
670   IF (T MOD 5)=0 THEN YS=4 ELSE YS=2
680   LINE (T*6+30,79-YS)-(T*6+30,79+YS),4
690 NEXT T
700 OPEN "GRP:" FOR OUTPUT AS #1
710 COLOR 4
720 FOR T=5 TO ML(MF) STEP 5
730   X=T*6+18 : Y=85
740   DRAW "BM=X;,=Y;"
750   PRINT #1,T
760 NEXT T
770 '
780 X=DF*6+30
790 DRAW "BM=X;,0C4M=X;,157"
800 DRAW "H5BM=X;,157E5"
810 X=X-8-DF : DRAW "BM=X;,160"
820 PRINT #1,STR$(MF);"月";STR$(DF);"日"
830 '
840 LINE (28,180)-(42,182),CP,BF
850 LINE (108,180)-(122,182),CS,BF
860 LINE (188,180)-(202,182),CI,BF
870 DRAW "BM50,178"  : PRINT #1,"ｼﾝﾀｲ"
880 DRAW "BM130,178" : PRINT #1,"ｶﾝｼﾞｮｳ"
890 DRAW "BM210,178" : PRINT #1,"ﾁｾｲ"
900 CLOSE #1
910 RETURN
920 '
930 LINE (X-1,Y-1)-(X+1,Y+1),C,BF
940 RETURN
950 '
960 '  コメント
970 '
980 CLS
990 PRINT NA$;" ｻﾝ" : PRINT
1000 PRINT "ｱﾅﾀﾉ ﾀﾝｼﾞｮｳ 日 ﾊ､";
1010 PRINT USING "####年 ##月 ##日";YB,MB,DB : PRINT
1020 PRINT "ﾖｿｳｽﾙ 日 ﾊ､ ";
1030 PRINT USING "####年 ##月 ##日";YF,MF,DF;
1040 PRINT "ﾃﾞｽﾈ｡":PRINT:PRINT
1050 PRINT "ｸﾞﾗﾌ ﾉﾅｶﾃﾞ､":PRINT
1060 PRINT "ｼﾝﾀｲ､ ｶﾝｼﾞｮｳ､ ﾁｾｲ ﾉ ｶｰﾌﾞｶﾞ":PRINT
1070 PRINT "ｿﾚｿﾞﾚ ﾏﾝﾅｶﾉ ｼﾞｸﾄﾏｼﾞﾜﾙ 日 ｶﾞ":PRINT
1080 PRINT "ﾖｳﾁｭｳｲ 日 ﾃﾞｽ !!":PRINT
1090 PRINT "ﾅﾆｶ ｷｰ ｦ ｵｽﾄ ｸﾞﾗﾌ ｶﾞﾃﾞﾏｽ｡ ";
1100 DM$=INPUT$(1) : RETURN
```

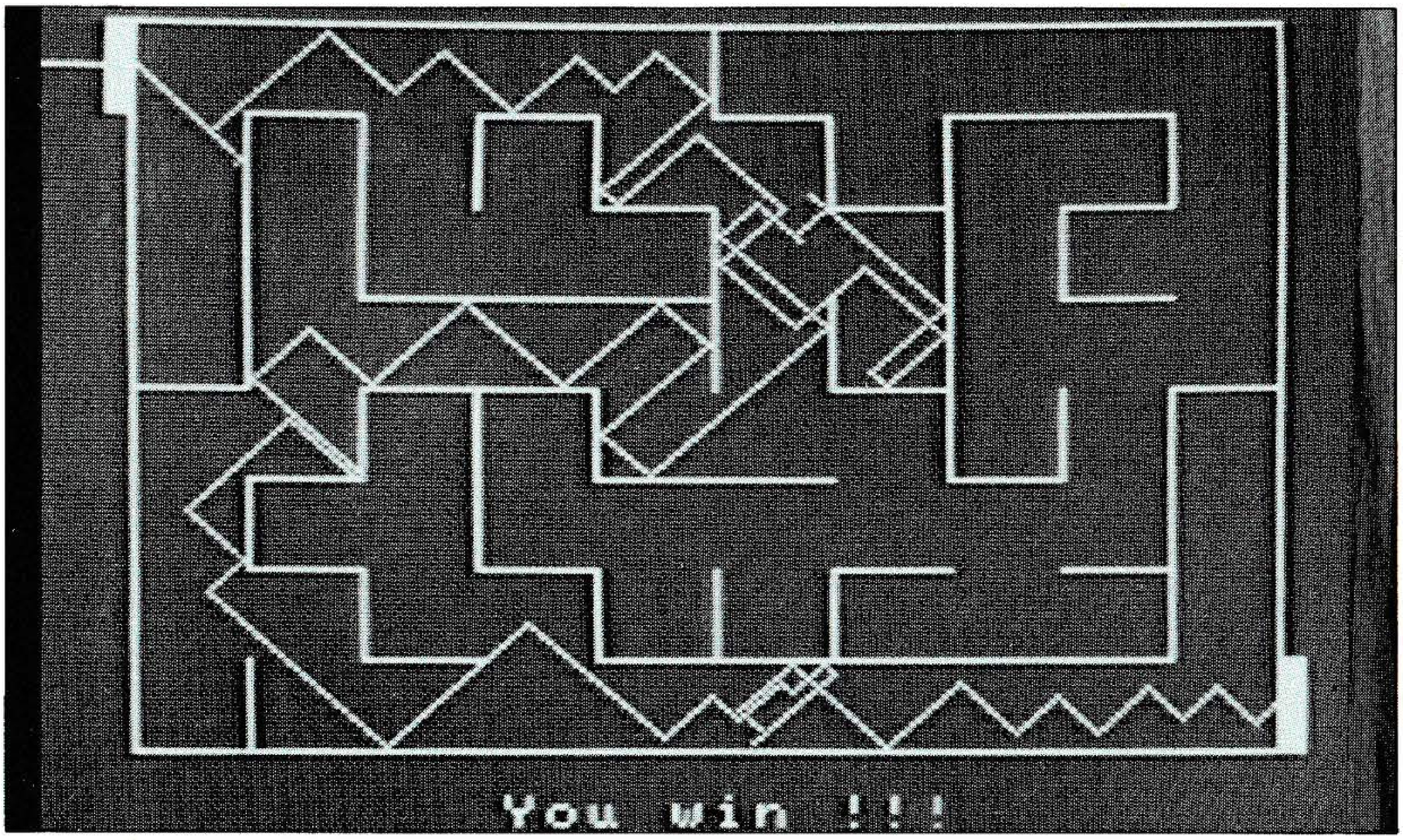
迷路ゲーム

4章に挙げられているプログラムは，ゲームのある1部分，1要素のみを取り出したものでした.「あんな画面はゲームセンターで見たことがあるぞ」というものもあったと思います．しかし，実際，自分でプログラムを組んでみようと思うと，いったいどうすればよいのか，まず何から始めればよいのか，よくわからないという人もいるでしょう．

そこで，実際に4章で取り上げた“4.7 迷路を作る”というプログラムをもとにして，ゲームを作ってみました．まず，これを参考にして，ゲームの作り方というものを，じっくりと研究してみてください．

利用法

まずゲームの遊び方を説明しておきます．このゲームは，迷路の入口から，常に斜めに移動する点を操作して，出口へぬけていくものです．その途中で点が壁にぶつかると点は反射し，ENERGY が1減ります．ENERGY は100から減っていきますが，これが0になったら終わりです. キー操作はテンキーを使い，右上-:，右下-／，左上-a，左下-zになっています（*Figure 1* 参照）．



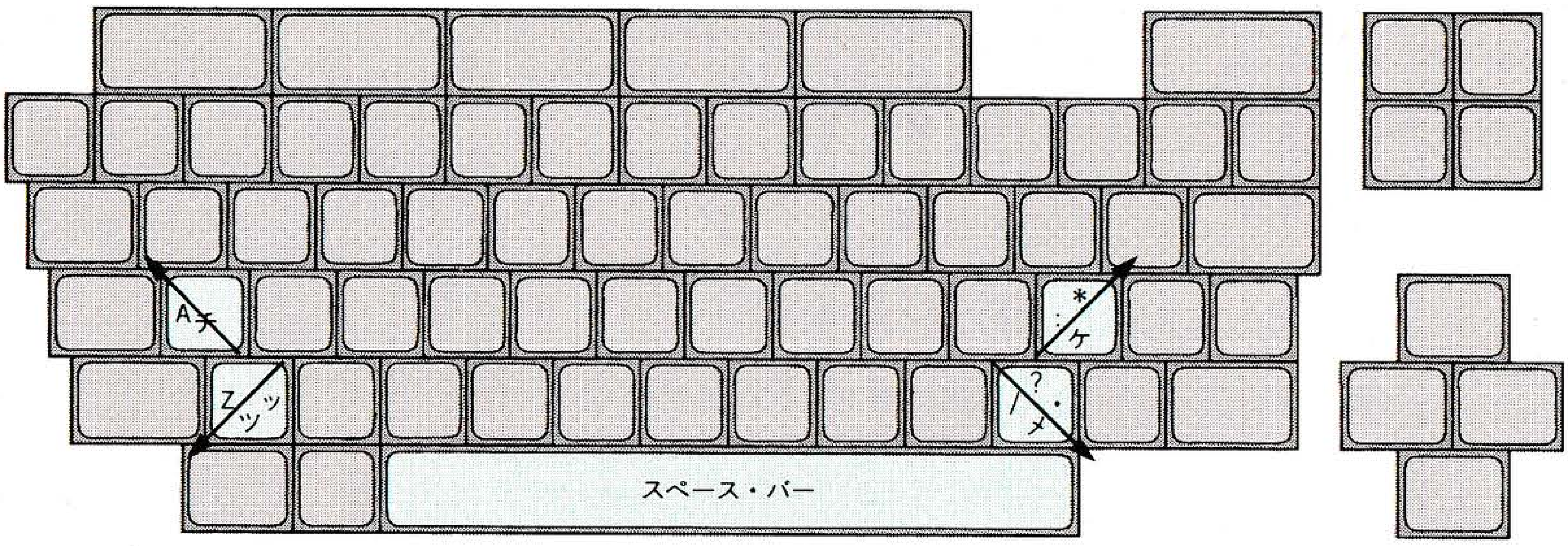


Figure 1

プログラムの説明

このプログラムは，前述の〝4.7 迷路を作る〟というプログラムがそのまま利用されていますが，それをうまく発展させたものです．

● 120行

ここでは，変数をすべて整数型にしています．このようにするメリットは，メモリを節約できる，プログラムの実行速度を速くすることができる，という2点です．

● 210〜270行

ここが，点を移動させていくためのメインの部分です．まず210行でキー入力を確かめ，キーが押されていたら，それに対応する移動量を配列 XS，YS から参照してきます．もしキー入力がない場合は，今まで点が移動してきた量 XS と YS を変更せずに，そのまま用います．

その値で，次に点を打つべき場所 XN，YN を決定し，240行でそこに壁があるかないかを判定します．そこに壁がない場合は，XD，YD にその座標を移して点を打ち，再びキー入力に戻ります．また，もし(XN，YN)に壁があるときには，実行は310行に移ります．

310行では，点が出口に当たったのかどうかを判断しています．出口だった場合には，1060行に飛んで，勝った場合の処理を行います．

320，330行では，点の進む先を調べて，横の壁があれば x 軸方向の移動量 XS の正負を逆に，縦の壁があれば y 軸方向の移動量 YS の正負を逆にしています．

340〜370行では壁に当たったということでエネルギーを示す変数 CT を1つ減らして表示し，音を出します．もしエネルギーが0になったら，ゲームは負けで，1110行以降の負けた場合の処理に実行が移ります．その他の場合は，再びメインの部分に戻ります．

● 420〜730行

これが，〝4.7 迷路を作る〟プログラムの本体です．このプログラムは4章で示したものに，420〜440行の初期設定と，730行の RETURN をつけ加えたことのほかは変更

していません．なお，いうまでもなくこれは，160行から呼び出されるサブルーチンです．

● 770〜1020行

ここに，今までに出てこなかったテクニックが登場しています．770〜840行に注目してください．これは，ゲームが始まってから点を移動させるための，キー入力のために必要な部分なのです．

あるキーを入力して，その方向に点を移動させる場合，

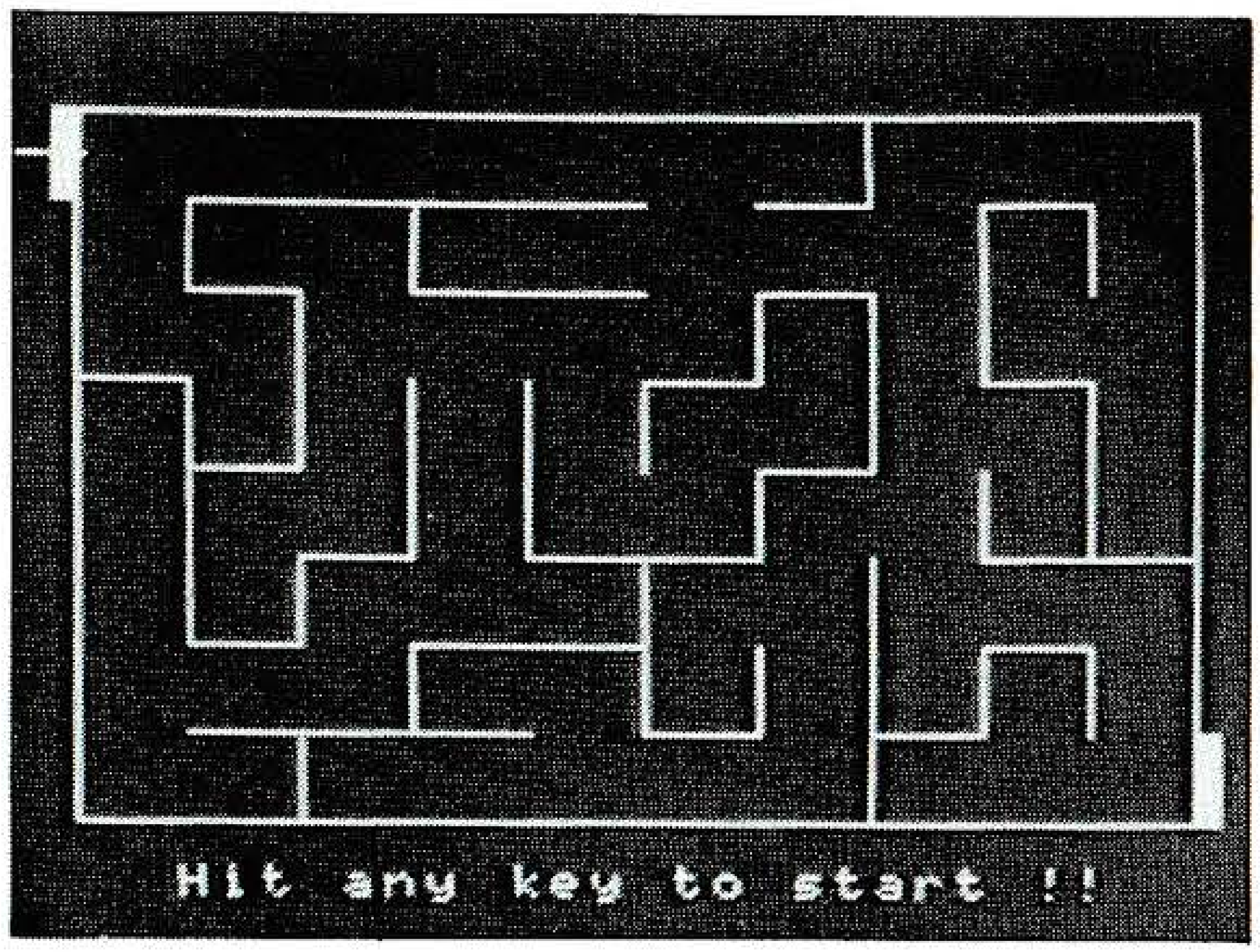
(1)キー入力を待つ
(2)そのキーを判断して動かす量を決定する
(3)その量の分だけ点を移動する

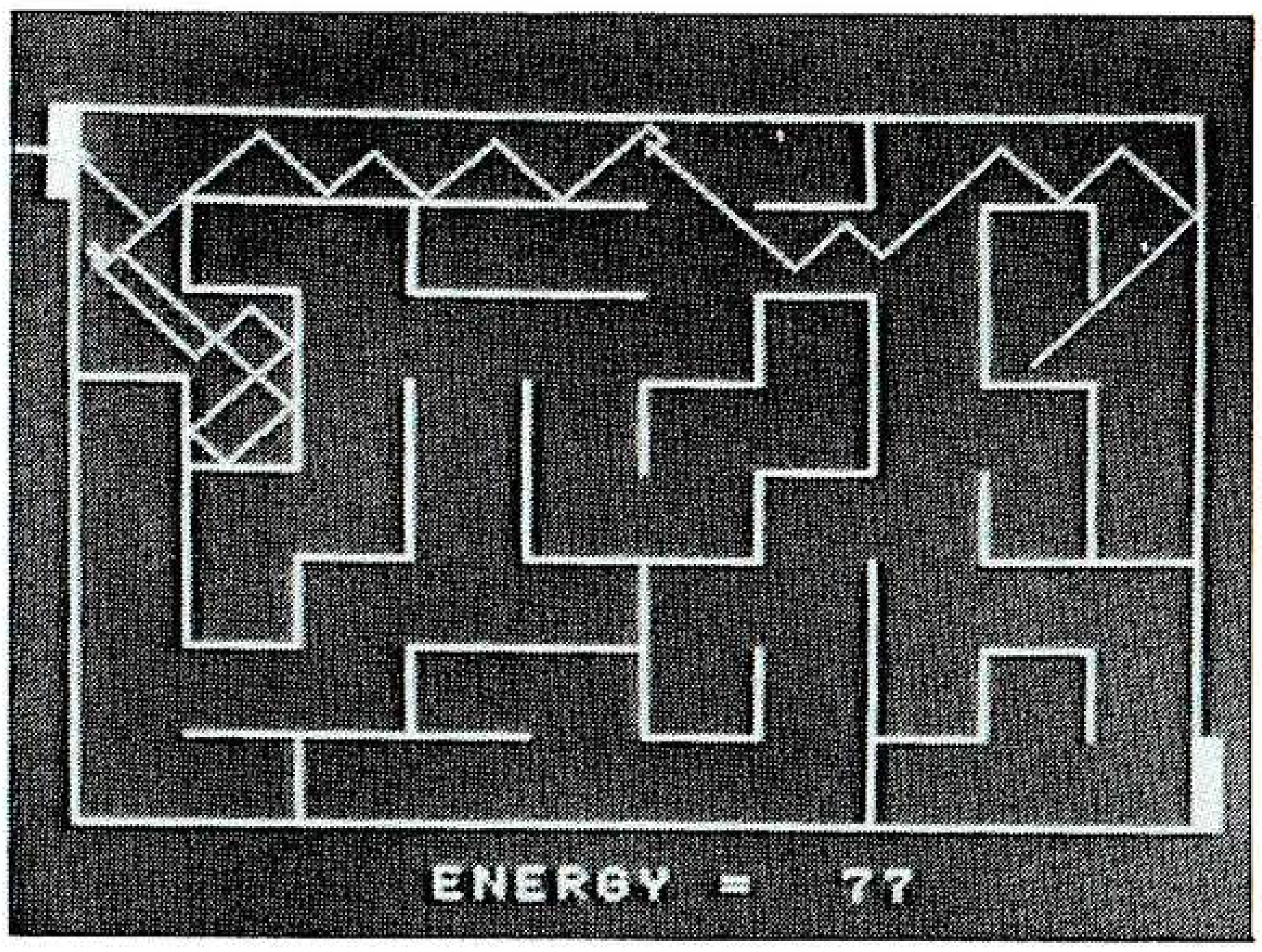
という3つの段階が考えられます．

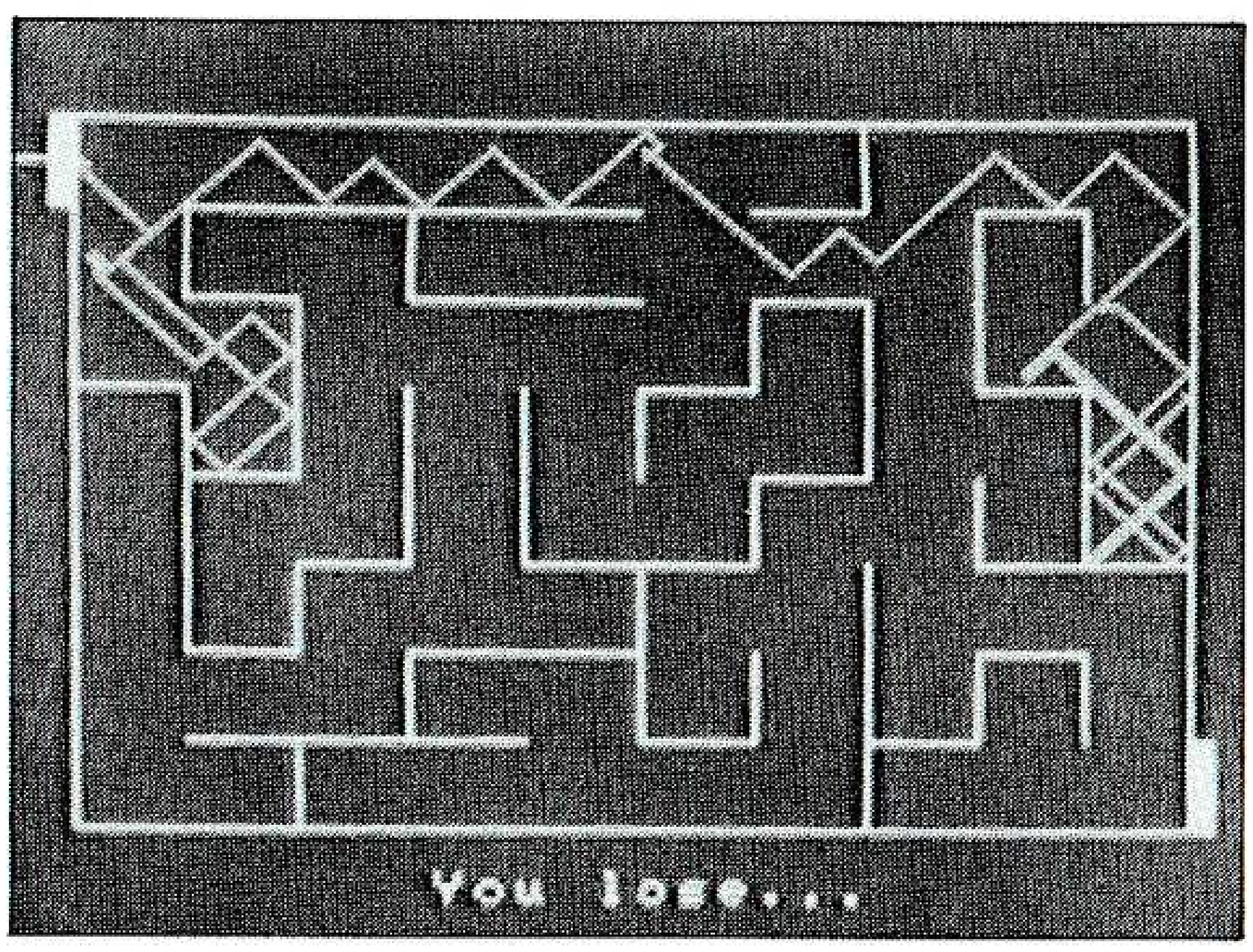
ここで，キー入力があるとすると，その入力されたキーに対するASCIIコードは，ASC（n$）という関数で得ることができます．このASCIIコードは，0〜255までが考えられますから，あらかじめ配列変数XSとYSを256個分用意し，キーが押されたとき，その配列変数XS，YSからその方向に対する移動量を引き出せるように，データを入れておくのです．もちろん，必要のないキーが押されたときには，そのASCIIコードに対応する配列変数XS，YSの値は，0になっています．

たとえば，文字“z”に対応するASCIIコードは，122です．キー“z”が入力されたとき，x軸方向に−2，y軸方向に1動かしたいとすると，あらかじめ，XS(122)=−2，YS(122)=1を定義しておかなければなりません．

もし，ここで，キー入力“z”があったとします．その“z”に対応する移動量は，







```
XS(ASC("z")),YS(ASC("z"))
```

で求められることがおわかりでしょう．そこで，あらかじめ XS（122）に−2，YS（122）に1を定義しておきます．つまり，準備には多少手間がかかるものの，キー入力がなされた段階では，すばやく処理することが，可能となるのです．

以上の準備をしている部分が，770〜840行なのです．

860〜940行は，点がスタート位置まで動いていき，そこでドアが描かれるという，アトラクションの部分です．

960〜980行は，何かキーが押されたらゲームを開始するという，待ち時間の部分です．

970行の

```
DM$=INPUT$(1)
```

は，何か1文字キー入力があるまで，そこで実行を止めておくために利用しています．

- 1060行〜1090行

ゲームに勝った場合の処理をする部分です．

- 1110行〜1150行

負けた場合の処理をする部分です．

キー入力のテクニック以外にはこれといって新しい方法は使っていませんが，なかなかおもしろいゲームになったでしょう．あなたも，他のプログラムを利用して，何かゲームを作ってみてください．はじめは，そのプログラムに，ほんの少し手を加えるだけでよいのですから！

List 2

```
100 ' List   2
110 '
120 DEFINT A-Z………変数をすべて整数型にする
130 BC=4 : PC=15
140 OPEN "GRP:" FOR OUTPUT AS #1
150 '
160 GOSUB 420
170 GOSUB 770
180 '
190 '  キー入力により点を移動するメインルーチン
200 '
210 I$=INKEY$
220   IF I$<>"" THEN XS=XS(ASC(I$)) : YS=YS(ASC(I$))
230   XN=XD+XS : YN=YD+YS
240   IF POINT(XN,YN)<>BC THEN GOSUB 310
250   XD=XN : YD=YN
260   PSET (XD,YD),PC
270 GOTO 210
```

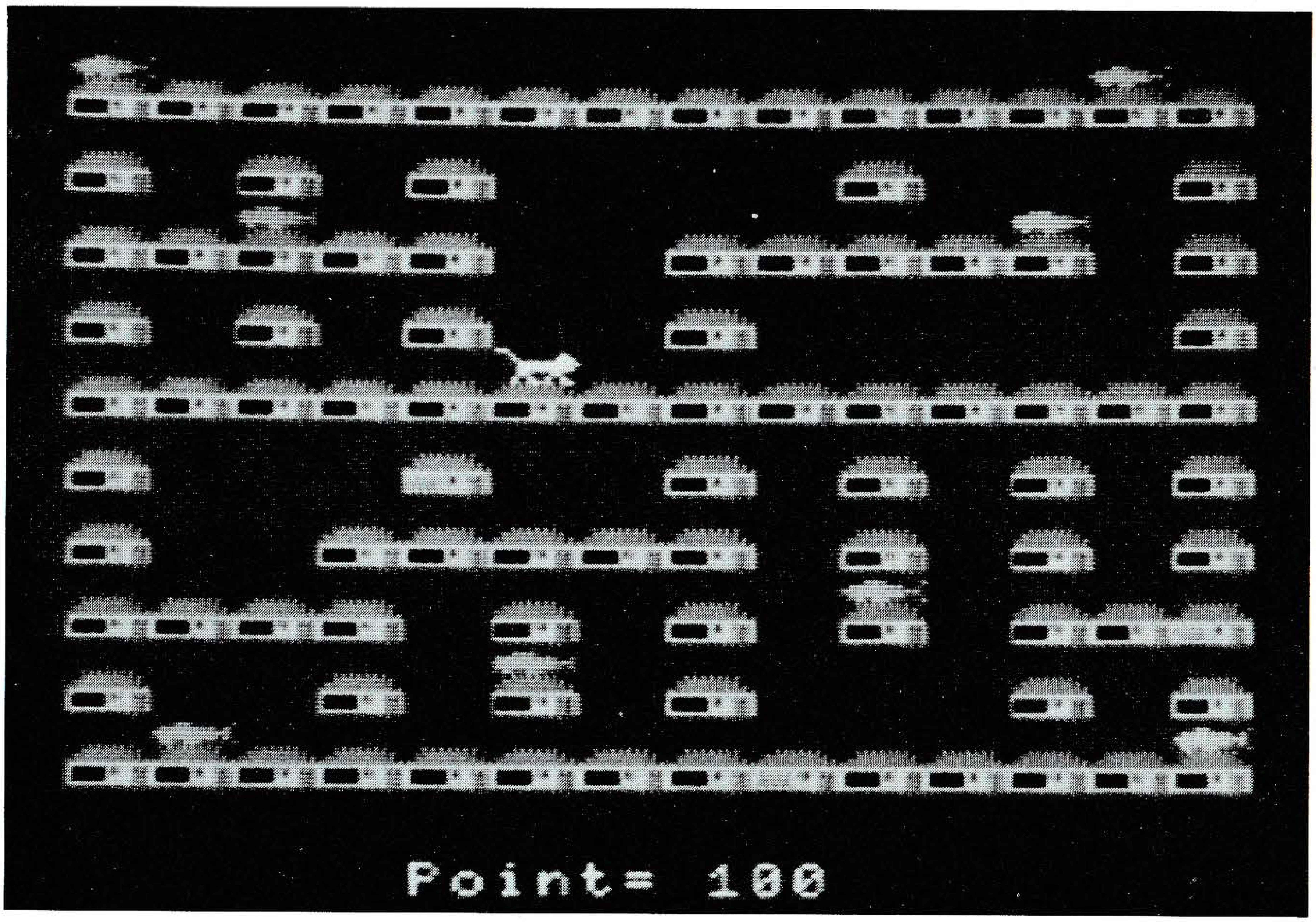
```
280 '
290 ' 壁に当たったかどうか判定するサブルーチン
300 '
310 IF XN>=XG AND YN>=YG THEN 1060
320 IF POINT (XN,YD)=PC THEN XS=-XS : XN=XN+2*XS
330 IF POINT (XD,YN)=PC THEN YS=-YS : YN=YN+2*YS
340 CT=CT-1 : PLAY "L32O6A"
350 LINE (162,180)-(210,187),BC,BF
360 DRAW "BM90,180" : PRINT #1,USING "ENERGY = ###";CT
370 IF CT=0 THEN 1110
380 RETURN
390 '
400 ' 迷路を作るサブルーチン
410 '
420 XB=20 : XE=240
430 YB=10 : YE=170
440 M=10  : N=8
450 S=(M-1)*(N-1)
460 XW=(XE-XB)/M
470 YW=(YE-YB)/N
480 DM=RND(-TIME)
490 C=PC
500 '
510 SCREEN 2 : COLOR 15,BC,BC : CLS
520 LINE (XB,YB)-(XE,YE),C,B
530 LINE (XB,YB+1)-(XB,YB+YW-1),BC
540 LINE (XE,YE-1)-(XE,YE-YW+1),BC
550 '
560 X=INT(RND(1)*(M+1)) : Y=INT(RND(1)*(N+1))
570   XO=XB+X*XW : YO=YB+Y*YW
580   IF POINT(XO,YO)=BC THEN 560
590   FOR I=1 TO 20
600     R=INT(RND(1)*4)
610     IF R=0 THEN XS=-1 : YS=0 : GOTO 650
620     IF R=1 THEN XS=1  : YS=0 : GOTO 650
630     IF R=2 THEN XS=0  : YS=1 : GOTO 650
640     IF R=3 THEN XS=0  : YS=-1
650     XN=XO+XS*XW : YN=YO+YS*YW
660     IF XN<=XB OR XN>=XE THEN 710
670     IF YN<=YB OR YN>=YE THEN 710
680     IF POINT(XN,YN)<>BC THEN 710
690     LINE (XO,YO)-(XN,YN),C
700     XO=XN : YO=YN : S=S-1
710   NEXT I
720 IF S>0 THEN 540
730 RETURN
```

```
740 '
750 '  ゲーム開始時の処理
760 '
770 DIM XS(255),YS(255)
780 RESTORE 830
790 FOR I=1 TO 6
800   READ A$,XS,YS
810   XS(ASC(A$))=XS : YS(ASC(A$))=YS      キー入力に際して必要なルーチン
820 NEXT I
830 DATA "a",-1,-1, "z",-1,1, ":",1,-1
840 DATA "/",1,1, "A",-1,-1, "Z",-1,1
850 '
860 XD=XB+1 : YD=YB+YW/2
870 FOR X=0 TO XB+1
880   PSET (X,YD),PC
890   PLAY "V12T200L32O5A"
900   FOR T=1 TO 100 : NEXT T
910 NEXT X
920 PLAY "L2O2A"
930 LINE (XB-5,YB)-(XB,YB+YW),PC,BF
940 LINE (XE,YE-YW)-(XE+5,YE),PC,BF      ゲームの初期画面
950 '
960 DRAW "BM40,180" : PRINT #1,"Hit any key to start !!"
970 DM$=INPUT$(1)
980 LINE (40,180)-(40+8*30,187),BC,BF
990 '      スコア(最高点)
1000 CT=100 : XG=XE : YG=YE-YW
1010 DRAW "BM90,180":PRINT #1,USING "ENERGY = ###";CT
1020 RETURN
1030 '
1040 '  ゲーム終了時の処理
1050 '
1060 PLAY "L4O4CL16FR16FR16L2F"
1070 LINE (90,180)-(210,187),BC,BF
1080 DRAW "BM90,180" : PRINT #1,"You win !!!"
1090 GOTO 1150
1100 '
1110 PLAY "L2O1AR4AR4A"
1120 LINE (90,180)-(210,187),BC,BF
1130 DRAW "BM90,180" : PRINT #1,"You lose..."
1140 '
1150 CLOSE #1:DM$=INPUT$(1):END
```

焼けてない トタン屋根の上のネコ

簡単なグラフィックを使ったゲームとして,「ネコが, 屋根から屋根へと飛び移り魚を取って行く」というゲームを考えてみました. しかし, それだけではおもしろくありません. そこで,「ネコの足音に気づいた人々が, 次々と部屋の明かりをつけてネコを追ってくる」ことにします.「あなたは, その追手を振り切って, すべての魚を取らなければなりません.」

このゲームは今まで出てきたテクニックを応用して作られています. じっくりとリストをながめて, ゲームの作り方を研究してください.



利用法

追いかけてくる部屋の明かりから逃げながら，魚を一尾取ると100点，一度でも捕まると点数は0になります．全部の魚を取るとゲーム終了，点数が表示されます．ネコを操作するにはカーソルキーを用います．ネコは，カーソルキーによって操ってください．なお，自ら明かりに向かっていった場合，うまく…….

もし，ジョイスティックを利用したい場合には，プログラム1830行のSTICK（0）をSTICK（1）もしくは，STICK（2）に変更してください．（STICK（1）かSTICK(2)かはそのジョイスティックのつながっているポートの番号で決まります．）

プログラムの説明

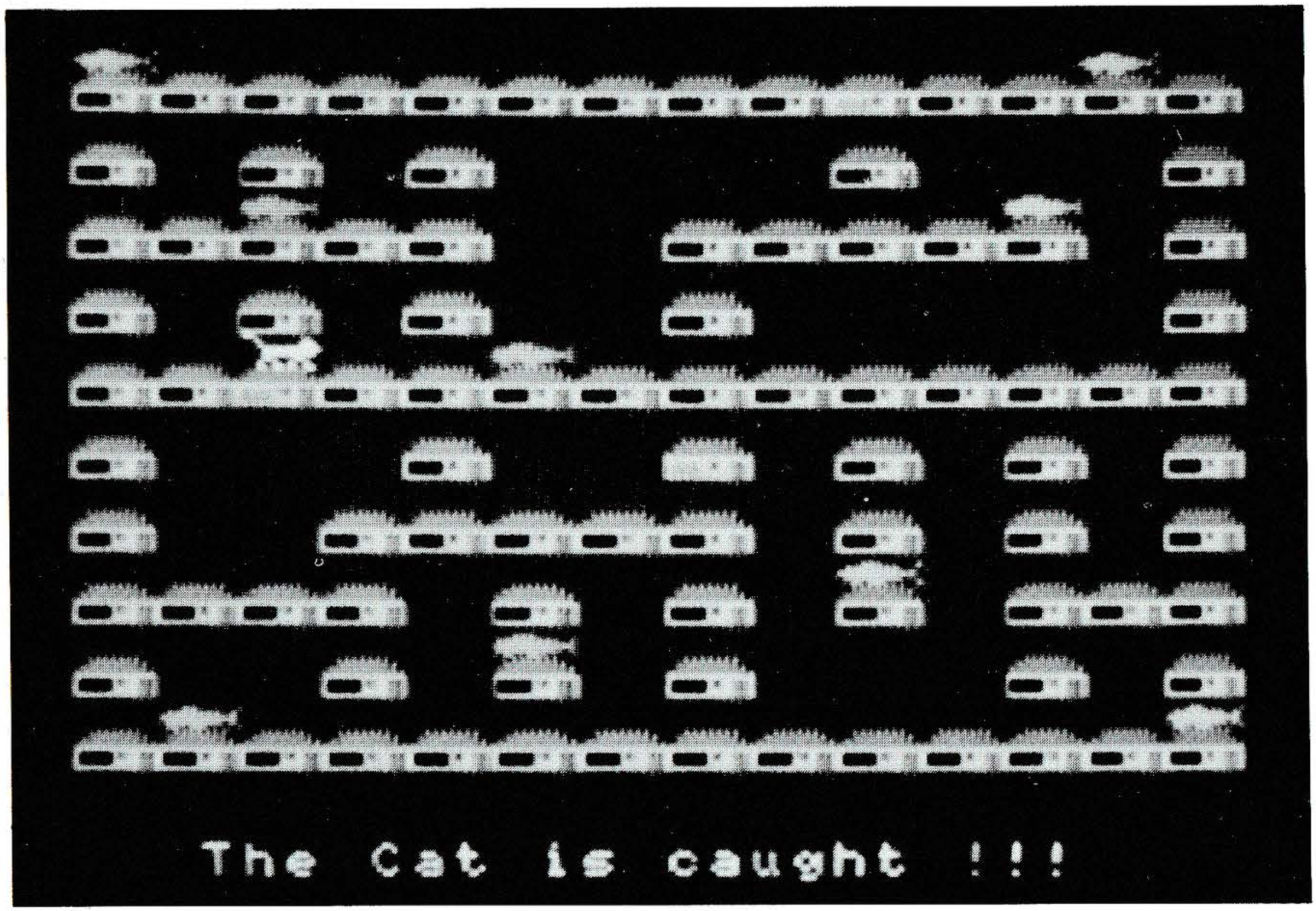
このプログラムは，ブロックごとに書かれているので，上から順に理解していってください．なお，肝心のグラフィックでは，ネコと家の明かりの部分にスプライト機能を利用しています．

- 120〜250行

  変数の定義，初期化をする部分です．

- 290〜370行

  右向きのネコの，スプライトパターン・



データを読み込んでいます．ここでは，16×16のスプライト機能を利用しています．このデータの作り方は，マニュアルや，Appendix の〝スプライトパターン・ジェネレータ〟を参考にしてください．なお，データの中に0が多いので，STRING$ (8, CHR$ (0)) を利用しています．

● 410〜490行

左向きのネコのスプライトパターン・データを読み込んでいます．

● 530〜610行

家の明かりをスプライトパターン・データとして読み込んでいます．

● 650〜860行

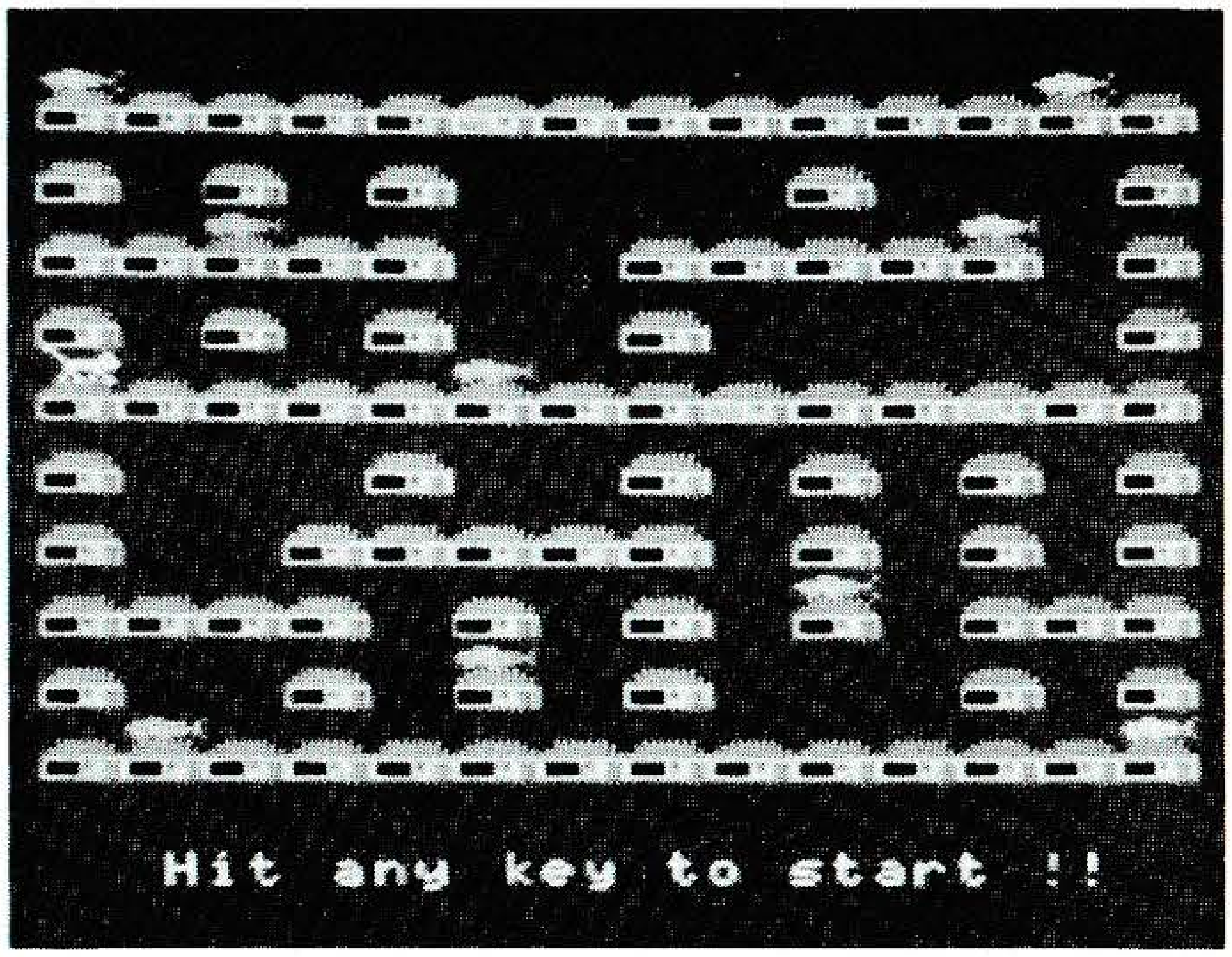
770〜860行までの町の中の地図データを，2次元配列文字変数 D$ (X , Y) に読み込みます．DATA の〝*〟は家のあるところ，〝1〟はその家の屋根に魚が乗っているところ，〝@〟は，ネコのはじめの位置を示しています．ここで，魚が何尾か，PC で数えています．この〝*〟，〝1〟，〝@〟は，横14，縦10のます目内であればどこへ動かしてもかまいません．770〜860行をいろいろと変えて，自分の好きな町を作り，ゲームを楽しんでください．

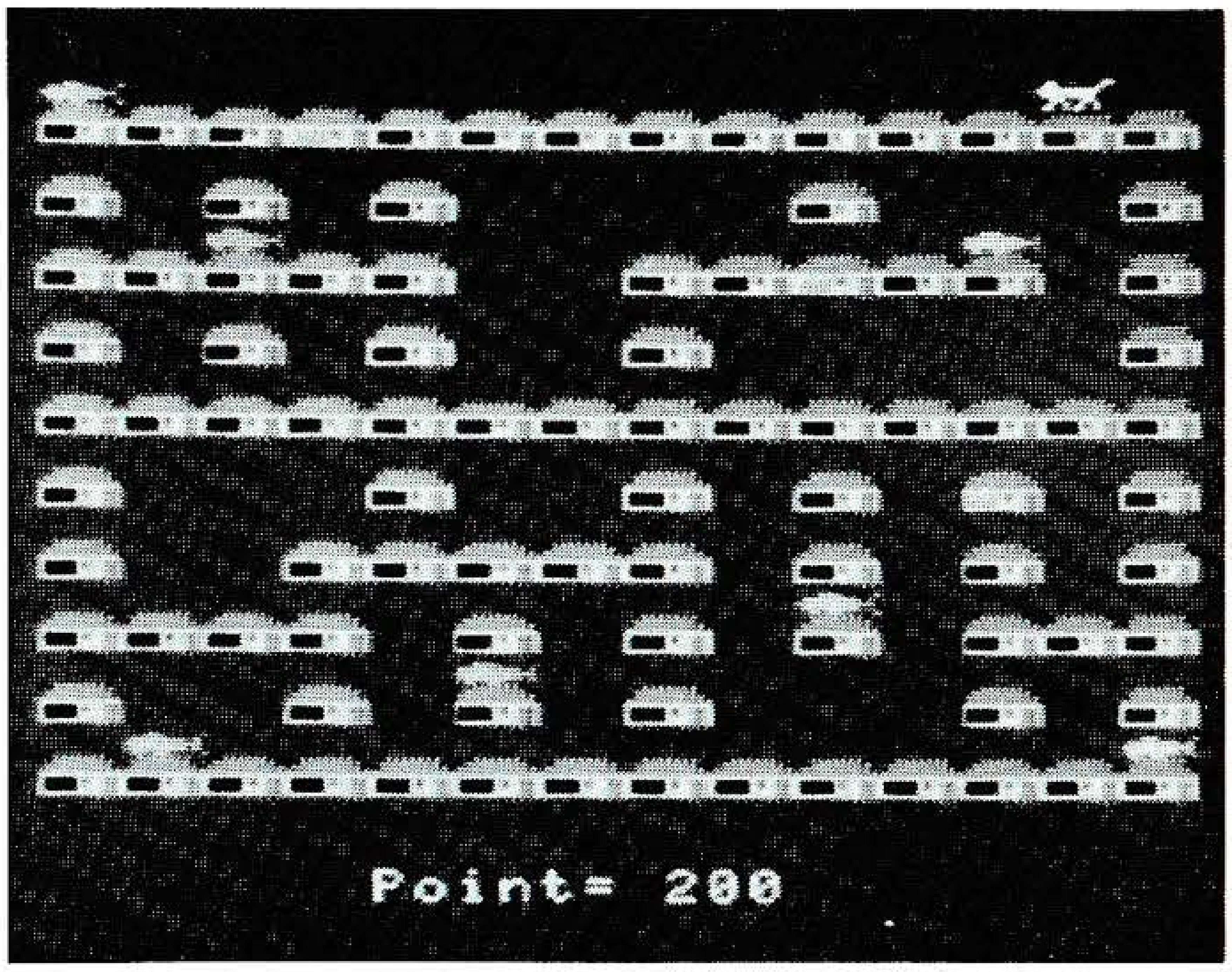
● 940，950行

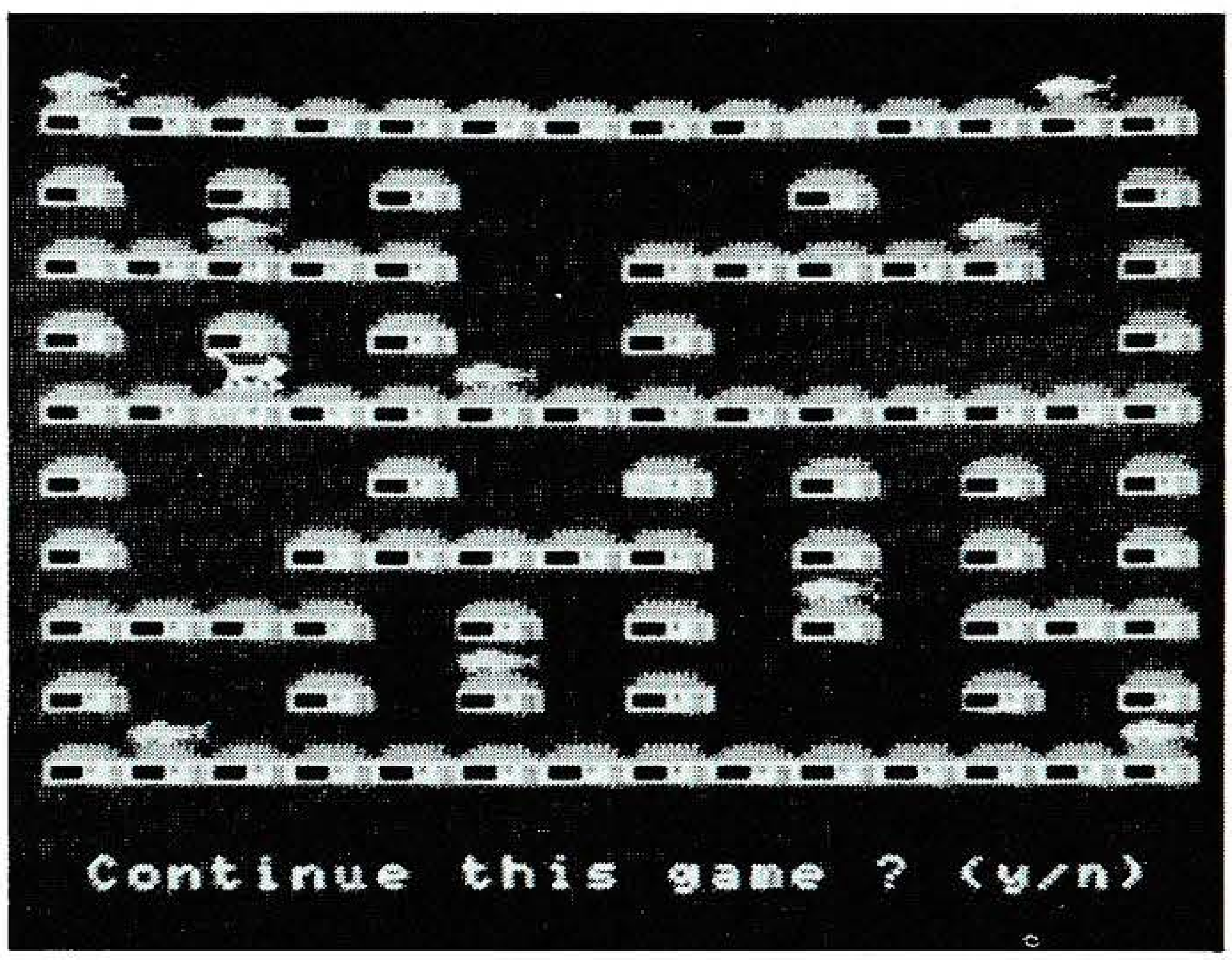
スプライトパターン・データを，スプライトパターン番号に割り当てます．0番に右向きのネコを，1番に左向きのネコを，2番に家の明かりのパターンを割り当てます．

● 990〜1060行

D$ (X , Y) の変数を参照して，画面に家と魚を出力します．配列変数の X は1〜14，Y は1〜10の値を取りますが，これに対応す







る画面の座標は，140行で定義されたDEF FN関数で計算されます．なお，前にも書きましたがグラフィック座標は，スプライト機能を出力するときに指定する座標と，縦に1ドット分ずれています．140行で定義した関数は，グラフィック用，スプライト用共に使いたいので，この1010行では，GY＝FNY(Y)+1とすることで，この座標のずれを修正しています．家や魚は1100〜1160行，1200〜1240行のサブルーチンで描かれます．

●1100〜1160行

座標(GX，GY)を基準に家を描くサブルーチンです．

●1280〜1350行

家のある所（ただし，ネコのいない所）を判断して(1300，1310行)，乱数でLM個の家に明かりをつけ，このデータを配列変数(LX(I)，LY(I))にたくわえておきます．この明かりが，ネコを追いかけるわけです．

●1390〜1450行

ネコの開始位置を示し，4回点滅させます．

●1470〜1520行

STOP 以外のキーを押すと，ゲームがスタートします．

●1560行

点数PTを0にしています．

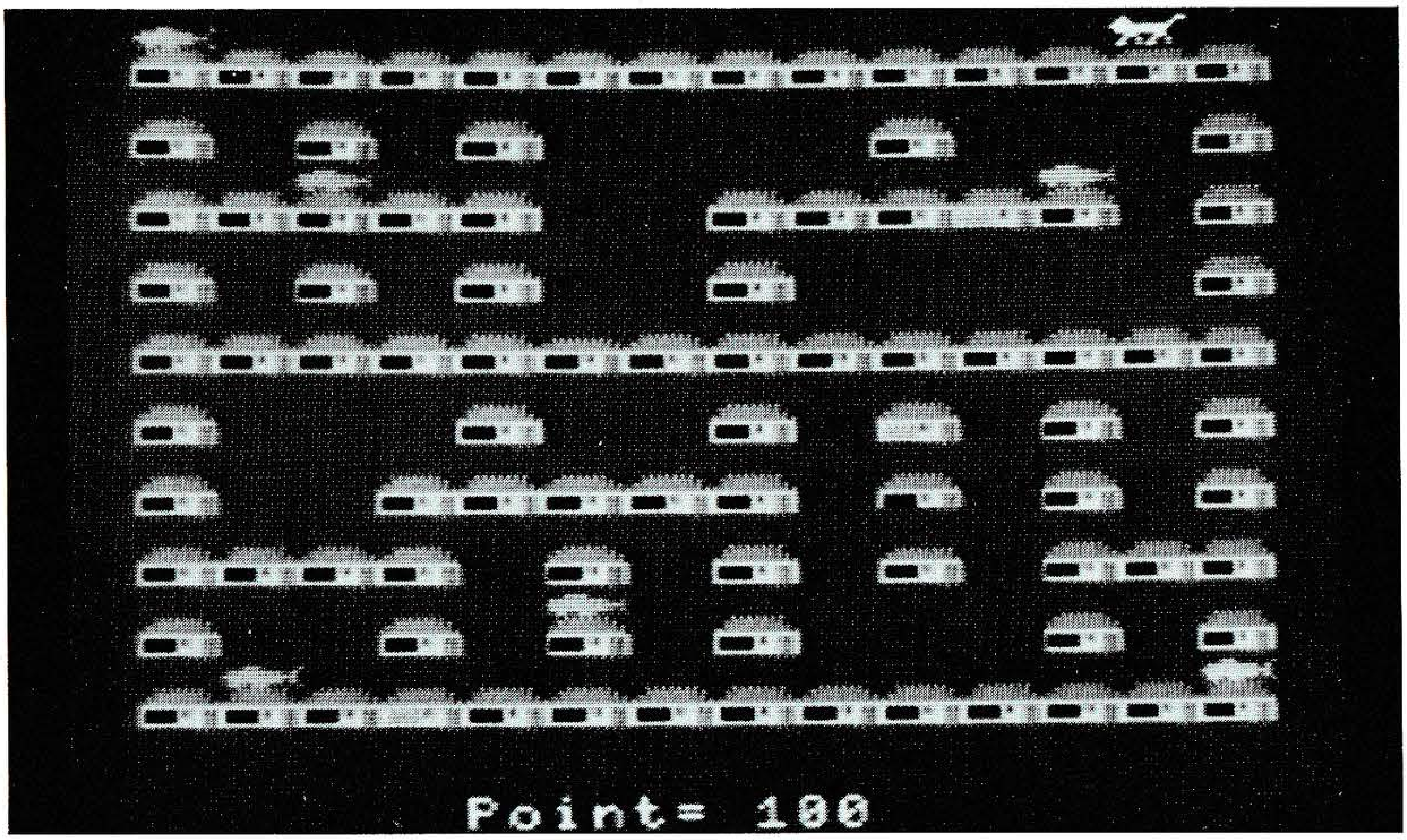
●1600行

"ピッ"という音を出しています．

●1620〜1790行

ここで，LM 個の明かり（この場合はLM＝3)が，ネコを追いかけています．この追いかけ方に注目してください．

1650行では，ネコが明かりよりも左にいれば7(←方向に対応する)，右にいれば3(→



方向に対応）という値を変数 RX に代入します．同様に1660行では，ネコが明かりよりも上にいるか下にいるかによって，1（[↑]）または5([↓])という値を変数 RY に代入します．RX，RY のどちら方向に進んでもネコに近づくことができるわけです．そして次の1670行で乱数によりこのうち一方を選び，変数 R に代入します．

このRの方向に移動したときの新しい場所を X，Y とします（1680行），そして，その位置に家があれば移動できるので，1750行へ進みます（1690行）．

もし，選んだ方向に家がなかった場合には，どこでもよいから進める方向を乱数で見つけ，R に代入し直します（1710～1730行）．また，LM 個ある明かりのうち最初の1つは，常にこの乱数だけで方向を決定するように1630行で判断しています．

明かりが動く方向が決定したら，1750～1780行で明かりを移動させます．このとき1780行で明かりとネコが重なったかどうか，判断しています．

● 1830～1860行

ネコを動かすためのキー入力を行っています．ただし，[↑] [→] [↓] [←]（それぞれ STICK 関数の1，3，5，7に対応）以外の方向は受けつけません．

● 1870～1910行

キー入力された方向に対して，ネコを動かします．しかし，その方向に家がない場合はカーソルキーの入力は無効になり，1600行へ戻ります．また，動く方向 R によって，右向き，左向きのネコを描き分けています（1890，1900行）．

1910行は，ネコが動いた位置に魚があるかないかを判断している部分です．魚があった場合，それをカウントして，魚を LINE 命令の Box fill で消しています（1960，1970行）．

● 2050～2090行

ネコが捕まった場合の処理をしている部分です．

● 2130～2210行

ネコが魚をすべて取った場合の処理です．

● 2230～2320行

再びゲームを続けるための処理です．

P.S.

このプログラム作成に協力してくださった，武田君，金子さんどうもありがとう．

List 3

```
100 ' List   3
110 '
120 DEFINT A-Z
130 DIM D$(15,11)
140 DEF FNX(X)=16*X : DEF FNY(Y)=16*(Y-1)-1
150 OPEN "GRP:" FOR OUTPUT AS #1
```

```
160 '
170 BC=12
180 LF=13 : WL=14 : DR=5 : LC=9
190 CC=15 : FC=7
200 '
210 LM=3 : DIM LX(LM),LY(LM)
220 XS(1)=0 : YS(1)=-1 :  XS(3)=1  : YS(3)=0
230 XS(5)=0 : YS(5)=1  :  XS(7)=-1 : YS(7)=0
240 '
250 DM=RND(-TIME)
260 '
270 ' 右向きネコをスプライトパターン・データとして定義
280 '
290 CR$="" : RESTORE 360
300 FOR J=1 TO 2
310   FOR I=1 TO 8
320     READ D : CR$=CR$+CHR$(D)
330   NEXT I
340   CR$=CR$+STRING$(8,CHR$(0))
350 NEXT J
360 DATA 96,16,8,7,7,6,9,9
370 DATA 0,4,14,255,254,24,36,166
380 '
390 ' 左向きネコをスプライトパターン・データとして定義
400 '
410 CL$="" : RESTORE 480
420 FOR J=1 TO 2
430   FOR I=1 TO 8
440     READ D : CL$=CL$+CHR$(D)
450   NEXT I
460   CL$=CL$+STRING$(8,CHR$(0))
470 NEXT J
480 DATA 32,112,255,127,24,36,101,0
490 DATA 6,8,240,224,96,144,144,0
500 '
510 ' 家の明かりをスプライトパターン・データとして定義
520 '
530 HL$="" : RESTORE 600
540 FOR J=1 TO 2
550   HL$=HL$+STRING$(8,CHR$(0))
560   FOR I=1 TO 8
570     READ D : HL$=HL$+CHR$(D)
580   NEXT I
590 NEXT J
600 DATA 0,0,0,0,126,126,126,0
610 DATA 0,0,0,0,0,0,0,0
```

```
620 '
630 ' 地図のデータを配列に取り込む
640 '
650 RESTORE 770
660 FOR Y=1 TO 10
670   READ D$
680   FOR X=1 TO 14
690     L$=MID$(D$,X,1)
700     IF L$="*" THEN D$(X,Y)="*"
710     IF L$="@" THEN D$(X,Y)="*" : CX=X : CY=Y
720     IF L$="1" THEN D$(X,Y)="1" : PC=PC+1
730   NEXT X
740 NEXT Y
750 '
760 '
770 DATA "1***********1*"
780 DATA "* * *    *   *"
790 DATA "**1**  ****1 *"
800 DATA "* * *  *     *"
810 DATA "@****1********"      地図のデータ
820 DATA "*   *  * * * *"
830 DATA "*  ***** * * *"
840 DATA "**** * * 1 ***"
850 DATA "*  * 1 *   * *"
860 DATA "*1***********1"
870 '
880 'This map is arranged by Mayumi Kaneko
890 '
900 ' スプライトパターン・データを文字列として定義
910 '
920 SCREEN 2,2 : COLOR 15,BC,BC : CLS
930 '
940 SPRITE$(0)=CR$ : SPRITE$(1)=CL$
950 SPRITE$(2)=HL$
960 '
970 ' 家と魚を表示
980 '
990 FOR Y=1 TO 10
1000   FOR X=1 TO 14
1010     GX=FNX(X) : GY=FNY(Y)+1
1020     IF D$(X,Y)="*" THEN GOSUB 1100
1030     IF D$(X,Y)="1" THEN GOSUB 1100 : GOSUB 1200
1040   NEXT X
1050 NEXT Y
1060 GOTO 1280
```

```
1070 '
1080 '  家を描くサブルーチン
1090 '
1100 DRAW "A0S4BM=GX;,=GY;BM+0,+10"
1110 DRAW "C=LF;E3R9F3L14U1R12U1L10"
1120 LINE (GX,GY+11)-STEP(15,4),WL,BF
1130 LINE (GX+1,GY+12)-STEP(5,2),1,BF
1140 LINE (GX+8,GY+12)-STEP(1,1),1,BF
1150 LINE (GX+12,GY+12)-STEP(2,3),DR,BF
1160 RETURN
1170 '
1180 '  魚を描くサブルーチン
1190 '
1200 DRAW "BM=GX;,=GY;BM+0,+4"
1210 DRAW "C=FC;E1R3U1R3D1R4D1R3U1E1"
1220 DRAW "BD4H1BL3L3D1L2U1L3D1L1U1L1H1"
1230 PAINT STEP(+3,0),FC
1240 RETURN
1250 '
1260 '  家の明かりを描く
1270 '
1280 FOR I=1 TO LM
1290   X=INT(RND(1)*14)+1 : Y=INT(RND(1)*10)+1
1300   IF D$(X,Y)="" THEN 1290
1310   IF X=CX AND Y=CY THEN 1290
1320   LX(I)=X : LY(I)=Y
1330   GX=FNX(X) : GY=FNY(Y)
1340   PUT SPRITE I,(GX,GY),LC,2
1350 NEXT I
1360 '
1370 '  ネコを描く
1380 '
1390 FOR I=1 TO 4
1400   GX=FNX(CX) : GY=FNY(CY)
1410   PUT SPRITE 0,(0,209),CC,0
1420   PLAY "T200V12L64O4A" : FOR T=1 TO 100 : NEXT T
1430   PUT SPRITE 0,(GX,GY),CC,0
1440   FOR T=1 TO 100 : NEXT T
1450 NEXT I
1460 '
1470 DRAW "BM40,176"
1480 PRINT #1,"Hit any key to start !!"
1490 DUMMY$=INPUT$(1)
1500 LINE (40,176)-(20+8*25,191),BC,BF
1510 DRAW "BM80,176"
1520 PRINT #1,USING "Point=####";PT
```

```
1530 '
1540 ' ゲームスタート
1550 '
1560 PT=0
1570 '
1580 ' ネコを追うためのサーチルーチン
1590 '
1600 PLAY "L32O3A"
1610 '
1620 FOR I=1 TO LM
1630   IF I<2 THEN 1710
1640 '
1650   IF CX<LX(I) THEN RX=7 ELSE RX=3
1660   IF CY<LY(I) THEN RY=1 ELSE RY=5
1670   IF RND(1)>.5 THEN R=RX ELSE R=RY
1680   X=LX(I)+XS(R) : Y=LY(I)+YS(R)
1690   IF D$(X,Y)<>"" THEN 1750
1700 '
1710   R=INT(RND(1)*4)*2+1
1720   X=LX(I)+XS(R) : Y=LY(I)+YS(R)
1730   IF D$(X,Y)="" THEN 1710
1740 '
1750   LX(I)=X : LY(I)=Y
1760   GX=FNX(X) : GY=FNY(Y)
1770   PUT SPRITE I,(GX,GY),LC,2
1780   IF X=CX AND Y=CY THEN 2050
1790 NEXT I
1800 '
1810 ' ネコを動かすためのルーチン
1820 '
1830 R=STICK(0)
1840 IF (R MOD 2)=0 THEN 1600
1850 X=CX+XS(R) : Y=CY+YS(R)
1860 IF D$(X,Y)="" THEN 1600
1870 CX=X : CY=Y
1880 GX=FNX(X) : GY=FNY(Y)
1890 IF R=1 OR R=3 THEN PUT SPRITE 0,(GX,GY),CC,0
1900 IF R=5 OR R=7 THEN PUT SPRITE 0,(GX,GY),CC,1
1910 IF D$(X,Y)<>"1" THEN 1600
```

```
1920 '
1930 ' ネコが魚を盗ったときのルーチン
1940 '
1950 PC=PC-1 : PT=PT+100 : PLAY "L16O5A"
1960 LINE (GX,GY)-STEP(15,8),BC,BF
1970 LINE (128,176)-(168,191),BC,BF
1980 DRAW "BM128,176"
1990 PRINT #1,USING "####";PT
2000 D$(X,Y)="*" : IF PC=0 THEN 2130
2010 GOTO 1600
2020 '
2030 ' ネコがつかまったときのルーチン
2040 '
2050 PLAY "L1O1A" : LINE (80,176)-(190,191),BC,BF
2060 DRAW "BM40,176"
2070 PRINT #1,"The Cat is caught !!!"
2080 FOR T=1 TO 2500 : NEXT T
2090 R$="L" : GOTO 2230
2100 '
2110 ' ネコが魚を全部盗ったときのルーチン
2120 '
2130 PLAY "L1O4A" : LINE (80,176)-(190,191),BC,BF
2140 DRAW "BM24,176"
2150 PRINT #1,"The Cat has eaten all fish !!"
2160 FOR T=1 TO 2500 : NEXT T
2170 LINE (0,176)-(255,191),BC,BF
2180 DRAW "BM72,176"
2190 PRINT #1,USING"Point=####";PT
2200 FOR T=1 TO 2500 : NEXT T
2210 R$="W" : GOTO 2230
2220 '
2230 LINE (0,176)-(255,191),BC,BF
2240 DRAW "BM12,176"
2250 PRINT #1,"Continue this game ? (y/n)"
2260 I$=INPUT$(1)
2270   IF I$="y" OR I$="Y" THEN LINE (0,176)-(255,191),BC,BF :
GOTO 2310
2280   IF I$="n" OR I$="N" THEN CLOSE #1 : END
2290 GOTO 2260
2300 '
2310 IF R$="L" THEN 1280
2320 IF R$="W" THEN 650
```

スキーゲーム

オセロがしたい，オセロがしたい，オセロがしたい，…….3回言ってみたところでオセロの盤もコマも相手も出てきやしません．そこで，神は我を見捨てたもうた！なんていってもしかたがなさそうなので，そこにあったマイコンでオセロゲームを作ることにしました．そして，完成したプログラムと対戦しているうちにメキメキメリと腕が折れ，じゃなかったメキメキと腕をあげ，この経験＋αを生かし，もっと強いオセロのプログラムを作りあげました．このオセロがオセロリーグで入賞したりするとマイコン症も本格的マイコン病へとご栄転．こんな人が，僕の友達に居るんですよ．

ゲームを通してオセロとプログラミングの2つを学んだのです．マイコンゲームも捨てたものではありません．

どうでしょう．あなたもゲームで遊びながら，スキーの腕に磨きをかけてみませんか．えっ，オセロとスキーをいっしょにするなって？　いえ，ところがこのスキーゲーム，重心のかけ方，タイミングのとり方，まさしく本物感覚のおすすめ品であります．スキー技術の向上もまちがいなし．

今年の冬のゲレンデは，あなたがスターかもしれない…….



利用法

プログラムを実行すると，まず画面の両側に1本，2本，……たくさんの木が表示されたのち，中央に1本，2本…10本程度の木が表示され，コースが完成します．次にスキーヤーがポッと現われたらゲームはいよいよスタートです．このスキーヤーこそあなた自身なのですよ．⇦⇨キーのどちらかを押してみてください．ほらエッジが斜面を捕え曲がったでしょう．木に激突しても，目を回したり，けがの心配は要りません，めげずにもう1度上から滑り直してください．初心者のためにもう一言，スペースバーを押してみてください．これぞ究極のボーゲンモードです．おためしあれ．

コースは全部で5つ，この5コースを50秒以内でクリアーすれば，ワールドカップの夢が見られることうけあい？

プログラムの説明

すべての木は，DRAW 命令と PAINT 命令で表示しています．コース中に立っている木には，葉の部分に重ねて，同じ緑色の小さなスプライトパターンを置いています(*Figure 1*)．これは，スキーヤーが，木に軽く触れたぐらいではそのまま通り抜け，激しく衝突したときに転倒と見なして滑り直しとしているからです．スキーヤーが木の中心に当たると小さなスプライトパターンが重なり，ON SPRITE GOSUB 命令による割り込みを発生し，滑り直しをさせられるわけです．では，コース外の木と衝突したときは，どうするのかというと，ゲレンデと森の境界をスキーヤーが越えたかどうかをチェックしているだけなのです(*Figure 2*参照)．

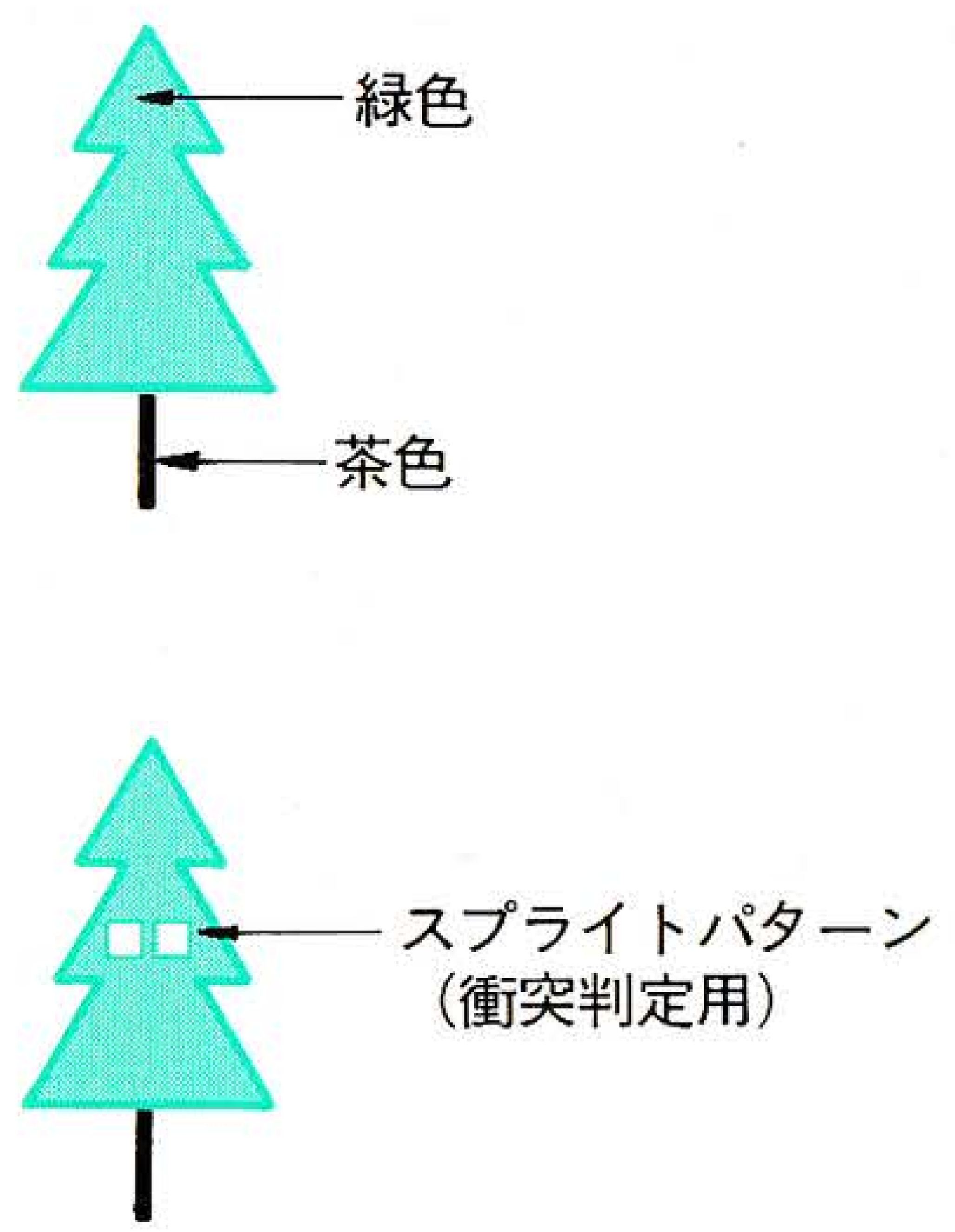


Figure 1

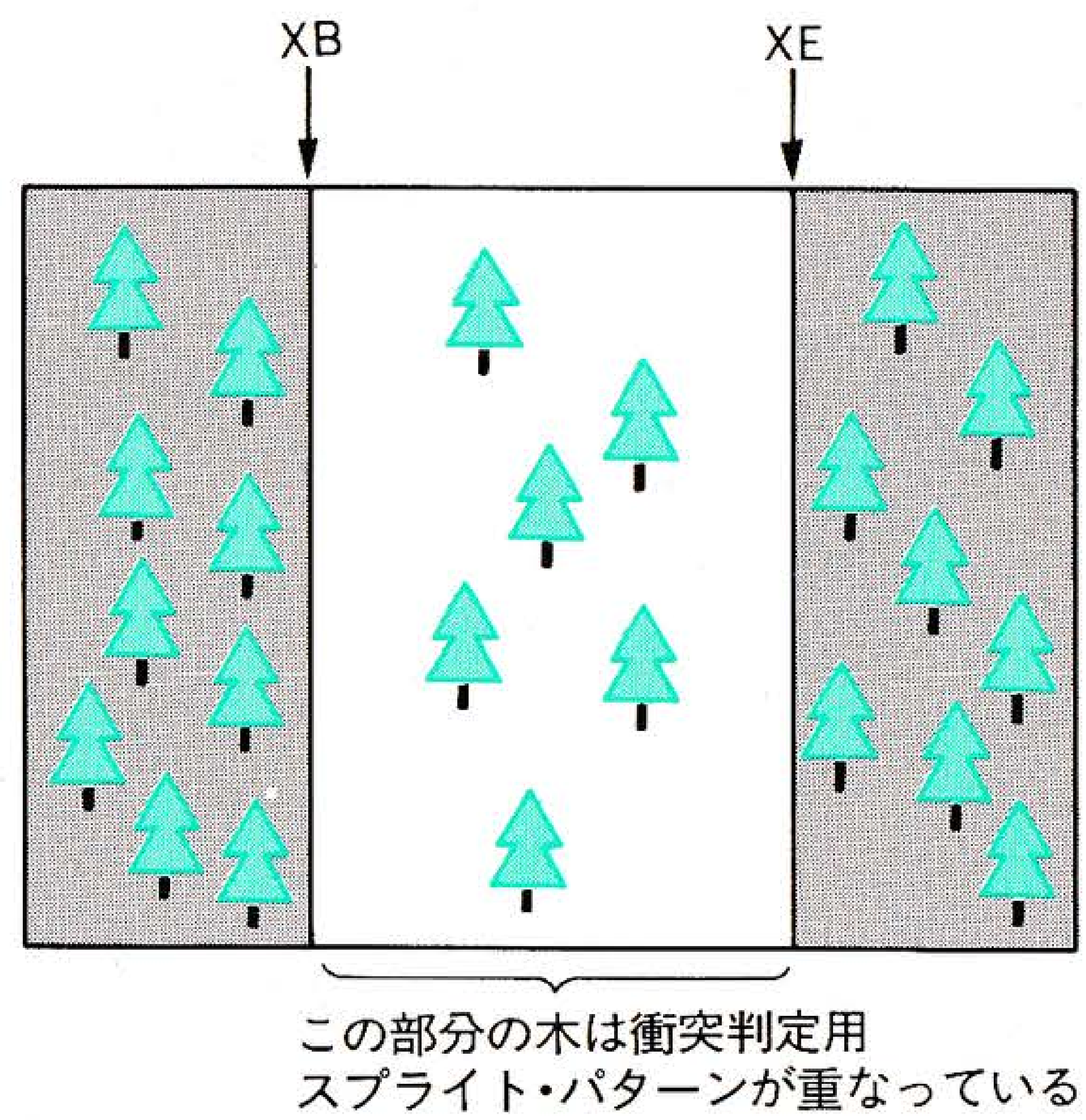


Figure 2

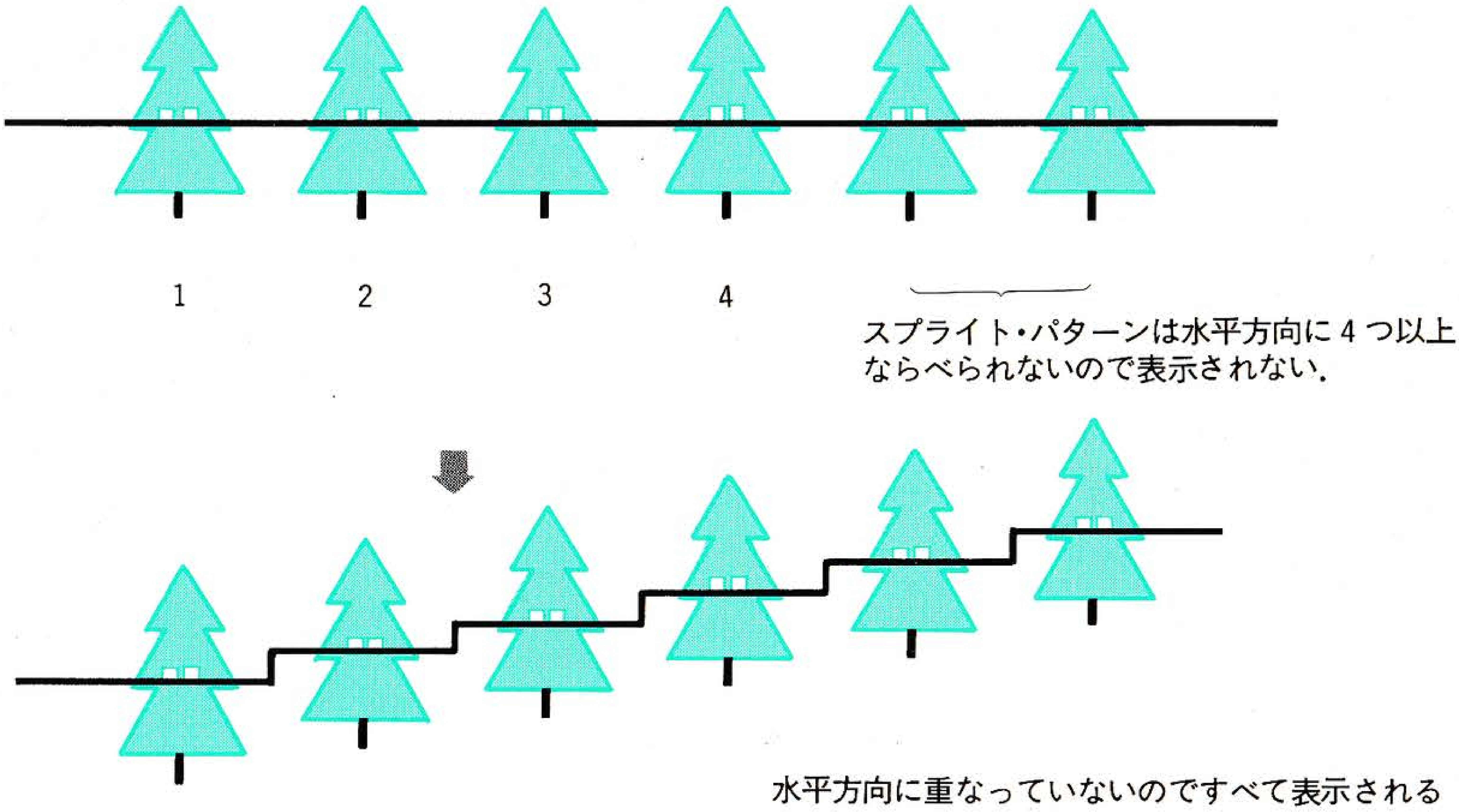


Figure 3

なぜコースの内と外で判断を変えるかといえば，スプライトパターンは，同時に32個しか表示できないので，全部の木に判定用の小さなスプライトパターンを置くことができないからです.

また，スプライトパターンを画面に表示したときに，水平に4つ以上パターンが並ぶと，それらは表示されません．そこで，このパターンを右へ行くほど上へずらして置くことで，水平方向に4つ以上パターンが重なることを防いでいます（*Figure 3* 参照).

それでは，実際にプログラムを見ていくことにしましょう.

● 120〜190行

変数の設定と初期化を行っています.

● 230〜300行

左向き，前向き，右向きの順にスキーヤーのスプライトパターンを定義しています．このスプライトパターン・データはAppendixのスプライトパターン・ジェネレータで作りました.

● 340〜360行

後に表示する木の中にかくしておく衝突判定用の小さなスプライトパターンを定義しています.

● 400〜430行

画面の両側に，森を描いています.

● 440行

ゲレンデのDATAを読み込むサブルーチン，そのデータに従って木を描くサブルーチンを呼びます.

● 460〜490行

DRAW命令とPAINT命令で，木を描くサブルーチンです.

● 530〜650行

中央のゲレンデに，データに基づいて木を表示します．この木は，スプライトパターンを重ねてあります．

● 670〜690行

ゲームが開始表示．

● 710行

スキーヤーが木にぶつかったとき，割り込みがかかるように，その処理行番号を定義しておきます．

● 730〜940行

カーソルキー入力とスペースバー入力によりスキーヤーを移動します．変数 XX，YY は加速度，XS，YS は速度，GX，GY は位置を示しています．これらの変数の処理によってスキーヤーの性格が決まりますが，ここではあまりむずかしいことは行っていません．特にプログラムテクニックというものはありませんから，省略．

● 920〜940行

1面をクリアーしたときの処理です．PP はゲレンデ番号を示す変数で，これが6になるとゲームは終わりです．

● 960〜1010行

スキーヤーが木に衝突したときスキーヤーの位置（GX，GY）を始めの位置にもどすサブルーチンです．

● 1030〜1130行

ゲームが一回終わったときの処理です．所要時間を表示し，何かキーが押されるまで待っています．ここで RETURN キーを押すとゲームは終了しますが，他のキーを押すと，ゲームが再び始まります．

● 1300行以降

ここからはゲレンデのデータです．5面分があります．この DATA 中の"＊"は，ゲレンデに木を表示する位置を表しています．その"＊"を並べかえると，自分だけのゲレンデが完成します．このとき，1つの面（10行分）に含まれる"＊"の数が，30個を越えないようにしてください．

初級，中級，上級コースなどを作っておくと楽しいでしょう．

List 4

```
100 ' List   4
110 '
120 SCREEN 2,2 : COLOR 1,15,15 : CLS
130 DEFINT A-Z
140 XB=60   : XE=190 : XW=(XE-XB)/10
150 GX=110  : GY=0   : SC=7  : X0=GX
160 XS=0    : YS=0   : XX=0  : YY=1
170 PP=1
180 '
190 DM=RND(-TIME)
```

```
200 '
210 ' スキーヤーをスプライトパターンとして定義
220 '
230 RESTORE 1150
240 FOR J=1 TO 3
250   S$=""
260   FOR I=1 TO 32
270     READ D : S$=S$+CHR$(D)
280   NEXT I
290   SPRITE$(J)=S$
300 NEXT J
310 '
320 ' 衝突判定用スプライトパターンを定義
330 '
340 TR$=STRING$(32,CHR$(0))
350 MID$(TR$,6)=STRING$(2,CHR$(5))
360 SPRITE$(0)=TR$
370 '
380 ' 森を描くルーチン
390 '
400 FOR I=1 TO 60
410   X=INT(RND(1)*(XB-12))      : Y=INT(RND(1)*175) : GOSUB 460
420   X=INT(RND(1)*(240-XE))+XE : Y=INT(RND(1)*175) : GOSUB 460
430 NEXT I
440 RESTORE 1310 : GOSUB 530 : GOTO 670
450 '
460 DRAW "S4A0 BM=X;,=Y;BM+7,+0C2"
470 DRAW "D1G2R1D1G2R1D2G2R1D1G2R4C6"
480 DRAW "D3U3C2R4H2R1H2U2R1H2U1R1"
490 PAINT STEP(-1,7),2 : RETURN
500 '
510 ' コースを作るルーチン
520 '
530 P=1
540 LINE (XB,0)-(XE,191),15,BF
550 FOR I=1 TO 10
560   READ D$
570   FOR J=1 TO LEN(D$)
580     IF MID$(D$,J,1)<>"*" THEN 630
590     X=(J-1)*XW+XB : Y=I*14+30-J*2
600     PUT SPRITE P,(X,Y),2,0 : GOSUB 460
610     PUT SPRITE P+1,(0,208),15,0
620     P=P+1 : IF P>30 THEN J=99 : I=99
630   NEXT J
640 NEXT I
650 RETURN
```

```
660 '
670 PUT SPRITE 0,(GX,GY),SC,2
680 PLAY "T255V12L4O4AR2.AR2.AR2.L2O5A"
690 FOR T=1 TO 2000 : NEXT T : TIME=0
700 '
710 ON SPRITE GOSUB 960 : SPRITE ON
720 '
730 I=STICK(0) : J=STRIG(0)
740 '
750   IF I=7 AND XS>=0 THEN XX=XX-4 : YS=YS/2 : GOTO 810
760   IF I=3 AND XS<=0 THEN XX=XX+4 : YS=YS/2 : GOTO 810
770   IF NOT(I=7 OR I=3) AND J=-1 THEN XS=XS/1.5 : YS=YS/2 : F=2
 : GOTO 830
780 '
790   XX=SGN(XS)*1 : YY=1
800 '
810   XS=XS+XX : YS=YS+YY
820   F=SGN(XS)+2
830   IF ABS(XS)>8 THEN XS=SGN(XS)*8
840   IF YS>8 THEN YS=8
850   GX=GX+XS : GY=GY+YS
860   IF GX<XB-8 OR GX>XE-7 THEN GOSUB 960
870   PUT SPRITE 0,(GX,GY),SC,F
880   PSET (GX+8,GY+8),14
890   IF J=-1 THEN PLAY "L64O2A" ELSE PLAY "L64O6A"
900 IF GY<191 THEN 730
910 '
920 SPRITE OFF : PP=PP+1 : X0=GX
930 IF PP>5 THEN 1030
940 XS=0 : XX=0 : GY=0 : GOSUB 530 : GOTO 710
950 '
960 SPRITE OFF
970 PLAY "L4O1AR4AR4A"
980 FOR T=1 TO 1000 : NEXT T
990 GX=X0 : GY=0 : XS=0 : YS=0 : XX=0 : YY=1
1000 PUT SPRITE 0,(GX,GY),SC,2
1010 RETURN 710
1020 '
1030 SPRITE OFF
1040 PLAY "T180O4L4GL8E.FL4GCL16DEFGL8FED"
1050 OPEN "GRP:" FOR OUTPUT AS #1
1060   DRAW "BM90,176"
1070   PRINT #1,USING"Time=####";TIME/60
1080 CLOSE #1
```

```
1090 I$=INKEY$ : IF I$<>"" THEN 1090
1100 I$=INPUT$(1)
1110 IF I$=CHR$(13) THEN COLOR 15,4,4 : END
1120 GX=110 : GY=0 : XS=0 : YS=0 : XX=0 : YY=1
1130 RESTORE 1310 : PP=1 : GOSUB 530 : GOTO 670
1140 'スプライトパターン・データ
1150 DATA 0,2,7,7,2,7,7,27
1160 DATA 11,6,2,6,7,12,49,198
1170 DATA 0,0,0,0,0,0,192,160
1180 DATA 160,160,209,206,124,98,128,0
```

左向きのスキーヤー

```
1190 '
1200 DATA 1,3,3,1,7,11,11,19
1210 DATA 43,74,10,12,12,4,4,4
1220 DATA 0,128,128,0,192,160,160,144
1230 DATA 168,164,160,96,96,64,64,64
```

右向きのスキーヤー

```
1240 '
1250 DATA 0,0,0,0,0,0,3,5
1260 DATA 5,5,139,115,62,70,1,0
1270 DATA 0,64,224,224,64,224,224,216
1280 DATA 208,96,64,96,224,48,140,99
```

まっすぐなスキーヤー

```
1290 '
1300 'no.1
1310 DATA "  *       "
1320 DATA "       *  "
1330 DATA "          "
1340 DATA "    *     "
1350 DATA "          "
1360 DATA "*        *"
1370 DATA "      *   "
1380 DATA " *        "
1390 DATA "          "
1400 DATA "    *   * "
```

コースNo.1のデータ

```
1410 '
1420 'no.2
1430 DATA "   *   *  "
1440 DATA "          "
1450 DATA "     *    "
1460 DATA "          "
1470 DATA "*       * "
1480 DATA "  *       "
1490 DATA "      *   "
1500 DATA "          "
1510 DATA "    *    *"
1520 DATA " *        "
```

コースNo.2のデータ

```
1530 '
1540 'no.3
1550 DATA "**  *    *"
1560 DATA "        * "
1570 DATA "   *  *   "
1580 DATA "          "
1590 DATA "  *  * *  "
1600 DATA "          "
1610 DATA "   *  *   "
1620 DATA " *      * "
1630 DATA "    *     "
1640 DATA "*     *  *"
1650 '
1660 'no.4
1670 DATA "     *    "
1680 DATA "          "
1690 DATA "****  ****"
1700 DATA "          "
1710 DATA "  *       "
1720 DATA "       *  "
1730 DATA "          "
1740 DATA "* ** ** **"
1750 DATA "          "
1760 DATA "          "
1770 '
1780 'no.5
1790 DATA "*  *  *  *"
1800 DATA "          "
1810 DATA "          "
1820 DATA "***  *  **"
1830 DATA "  *     * "
1840 DATA "  *     * "
1850 DATA "*  *     *"
1860 DATA "     *    "
1870 DATA "          "
1880 DATA " ** ** ** "
```

コースNo.3のデータ

コースNo.4のデータ

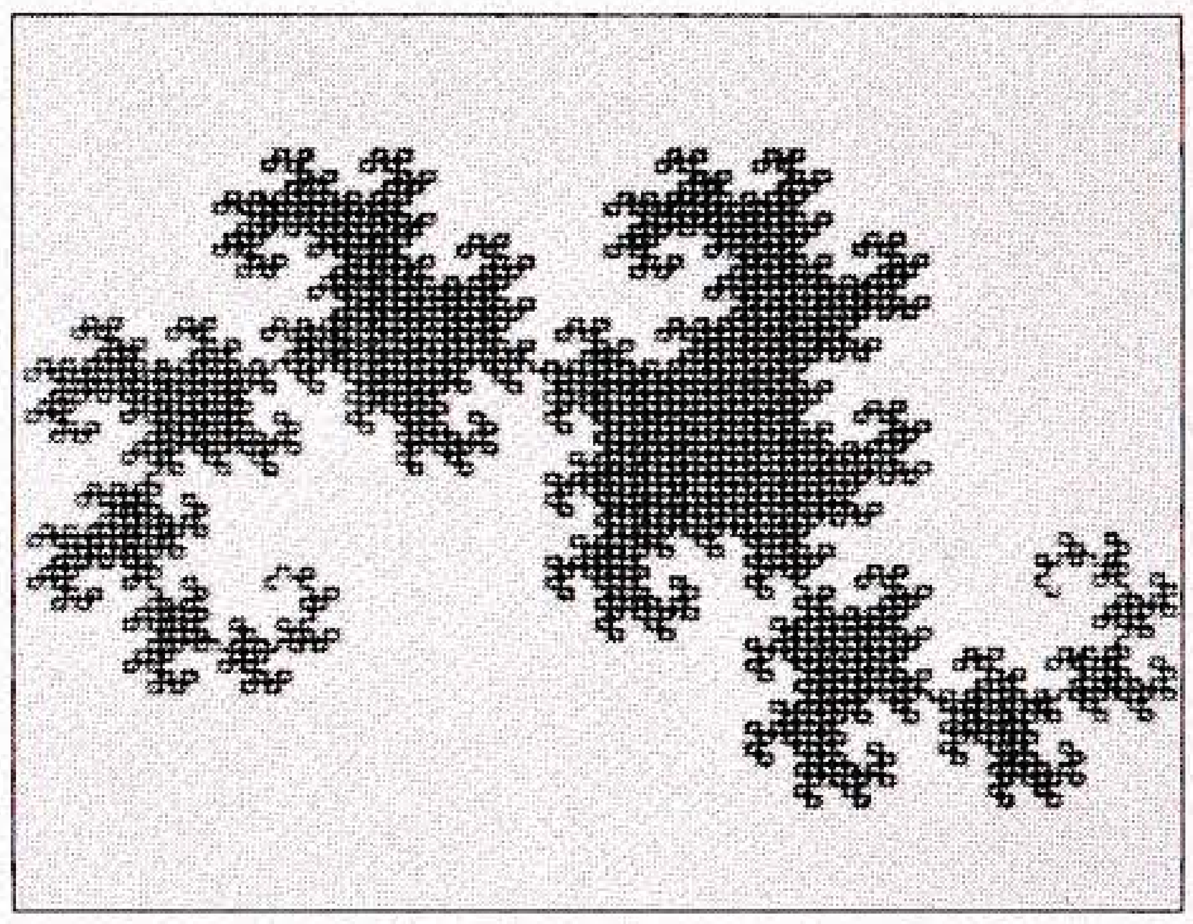
コースNo.5のデータ

ドラゴン曲線

97ページで再帰的なグラフィック図形の例をひとつご紹介しましたが，かのプログラムは，サブルーチンが自分自身のみをコールするという構造をとっているため，再帰といってもまだ理解しやすい部類に入ります．けれども世の中には，あるサブルーチンが別のサブルーチンをコールし，そのサブルーチンが別のサブルーチンをコールし，巡りめぐってまた自分自身が使われるという，ほとんど人間の理解のレベルを超越した構造の再帰的グラフィックが存在しています．

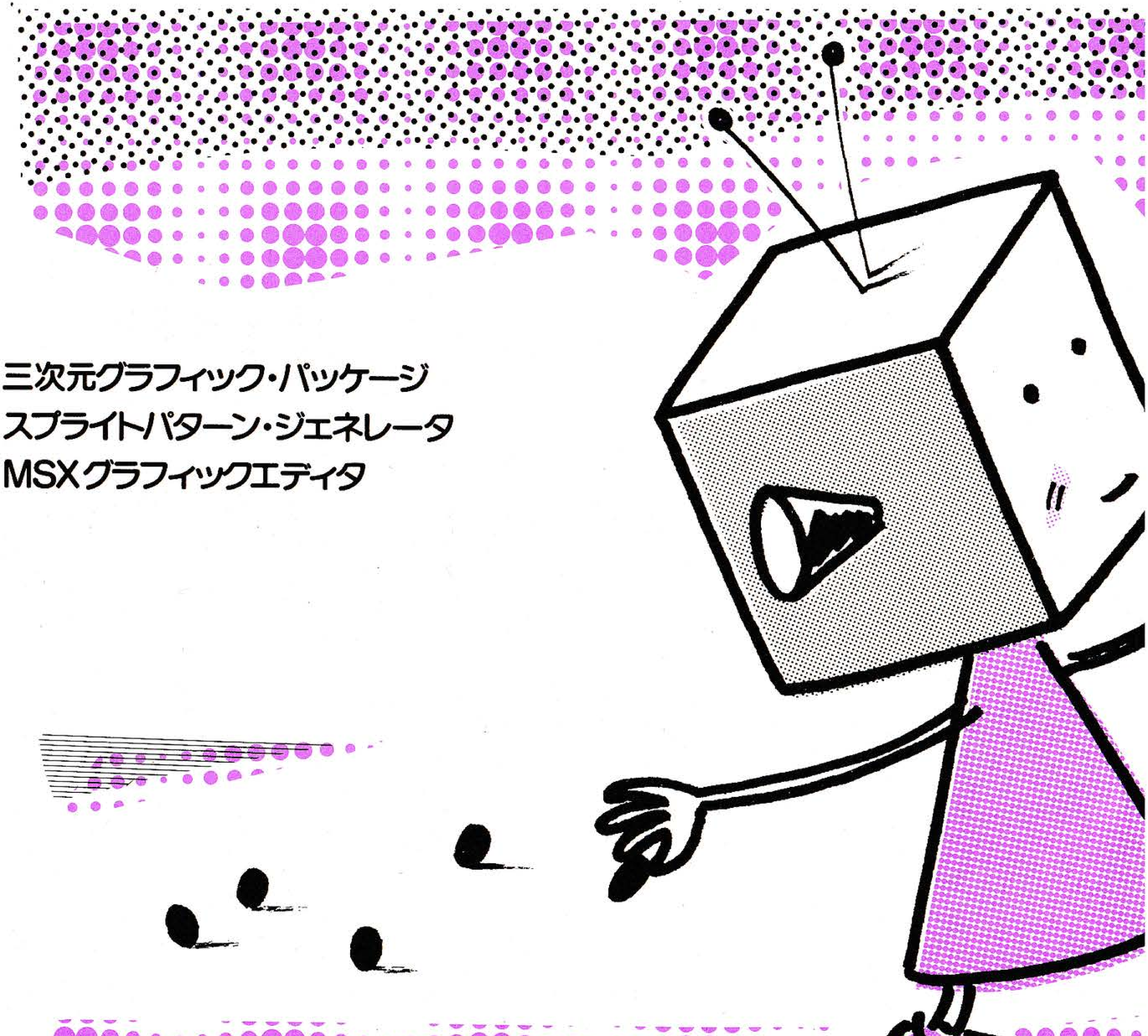
ここに掲載したのも，そのようなプログラムのひとつです．もっぱら左へ曲がりたがるサブルーチン(200行)と，逆に右へばかり曲がろうとするサブルーチン(280行)が，お互いに相手をコールしながらもつれあい，ある曲線を織り上げていきます．しかし見て頂けるとおわかりのように，これだけ複雑でありながら，同じ線の上を二度通ることは絶対にありません．この曲線はドラゴンカーブと呼ばれ，この手のグラフィックがすきな人々の間ではとても有名なシロモノです．

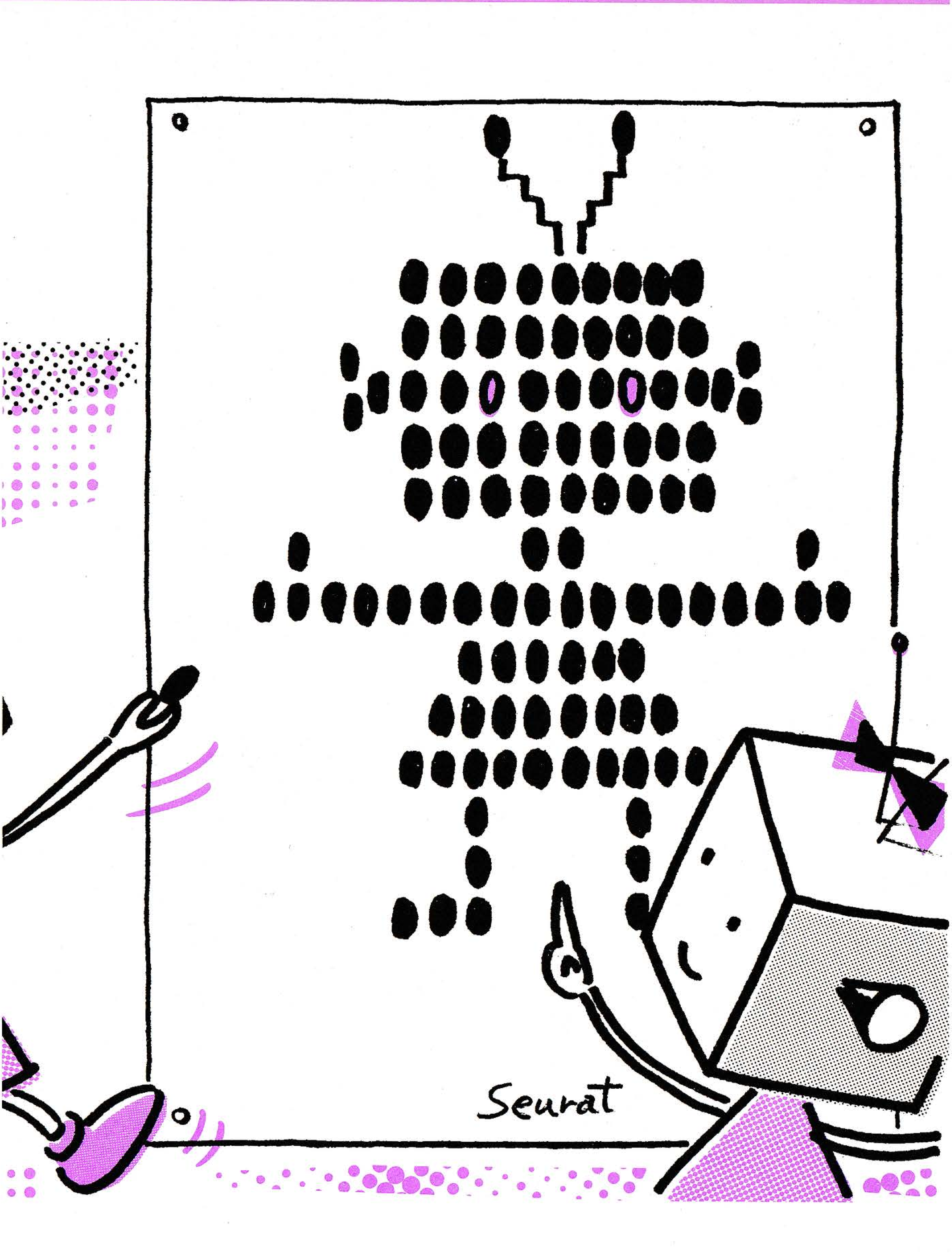
```
100 ' List Ex.5
110 '
120 SCREEN 2 : COLOR 1,15,15 : CLS
130 DEFINT A-Z
140 '
150 C=1
160 DRAW "BM70,120 A1S8 C=C;"
170   R=1 : D=12 : GOSUB 200 : C=16-C
180 GOTO 160
190 '
200 IF D=0 THEN DRAW "U" : RETURN
210   D=D-1
220   GOSUB 200
230   R=(R+3) MOD 4 : DRAW "A=R;"
240   GOSUB 280
250   D=D+1
260 RETURN
270 '
280 IF D=0 THEN DRAW "U" : RETURN
290   D=D-1
300   GOSUB 200
310   R=(R+1) MOD 4 : DRAW "A=R;"
320   GOSUB 280
330   D=D+1
340 RETURN
```



Appendix

三次元グラフィック・パッケージ
スプライトパターン・ジェネレータ
MSXグラフィックエディタ




Seurat

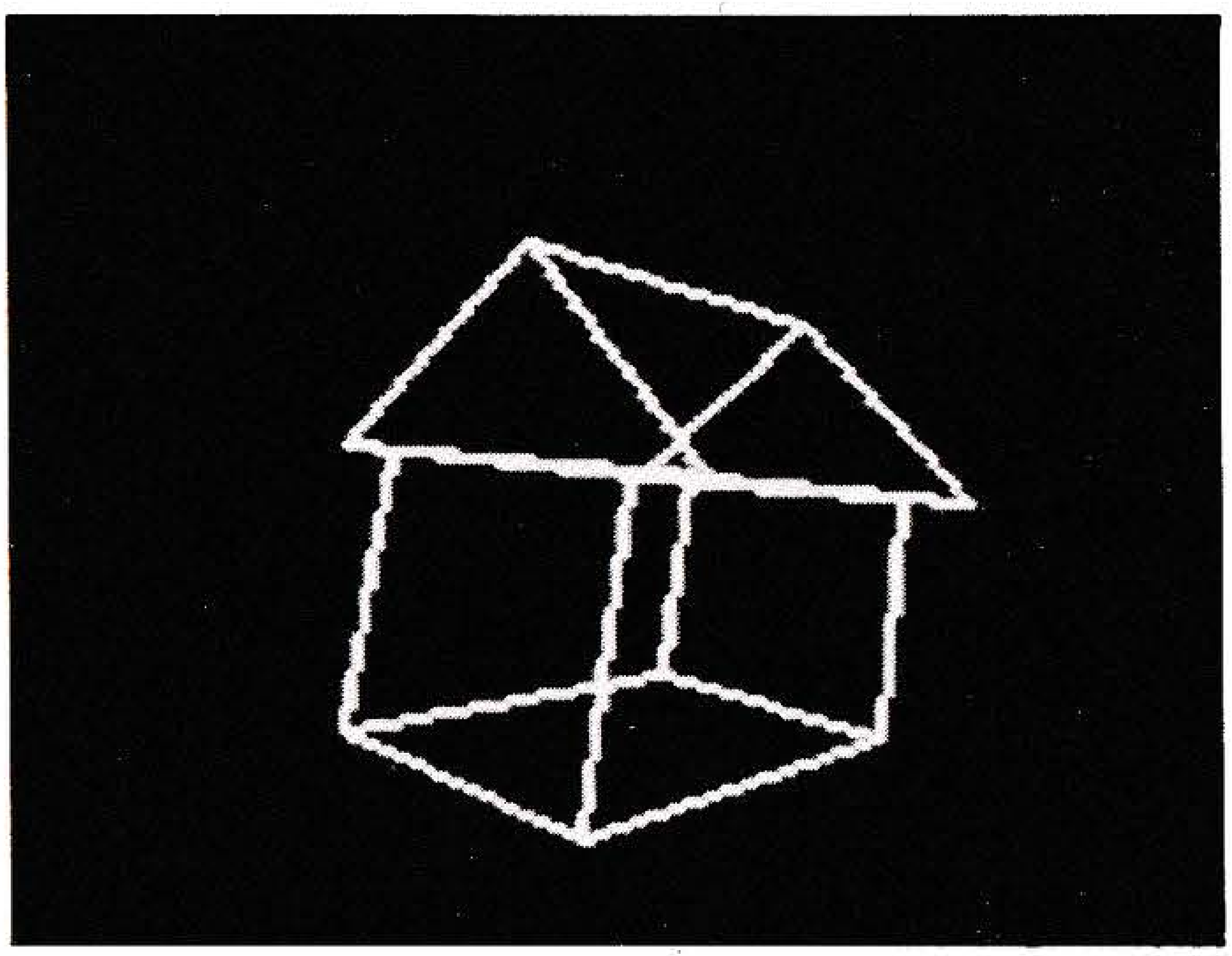
三次元グラフィック パッケージ

このプログラムは，立体のデータを自由に定義できるようになっています．サンプルとして，家の形を表すデータがついていますから，様々な方向からながめてみてください．

私たちの住む世界は，三次元と言われるように，すべてが立体からできています．もし，この立体をコンピュータにインプットし，それを好きな方向からながめられたら，どんなにすてきなことでしょう．それを実現するのが，この三次元グラフィック・パッケージです．

利用法

3次元空間は，*Figure 1*のように3次元座標(x，y，z)で表されます．図形は原則



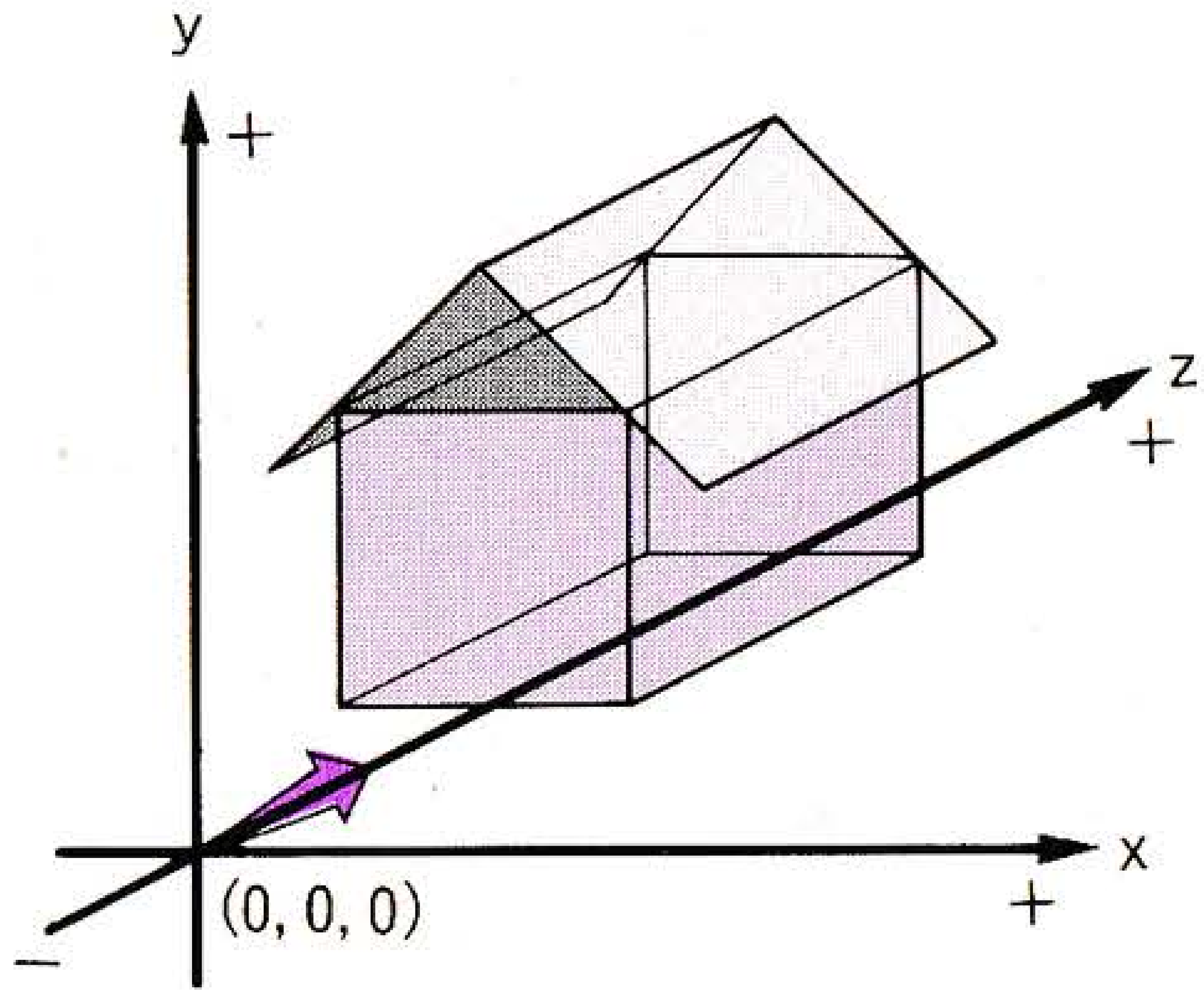


Figure 1

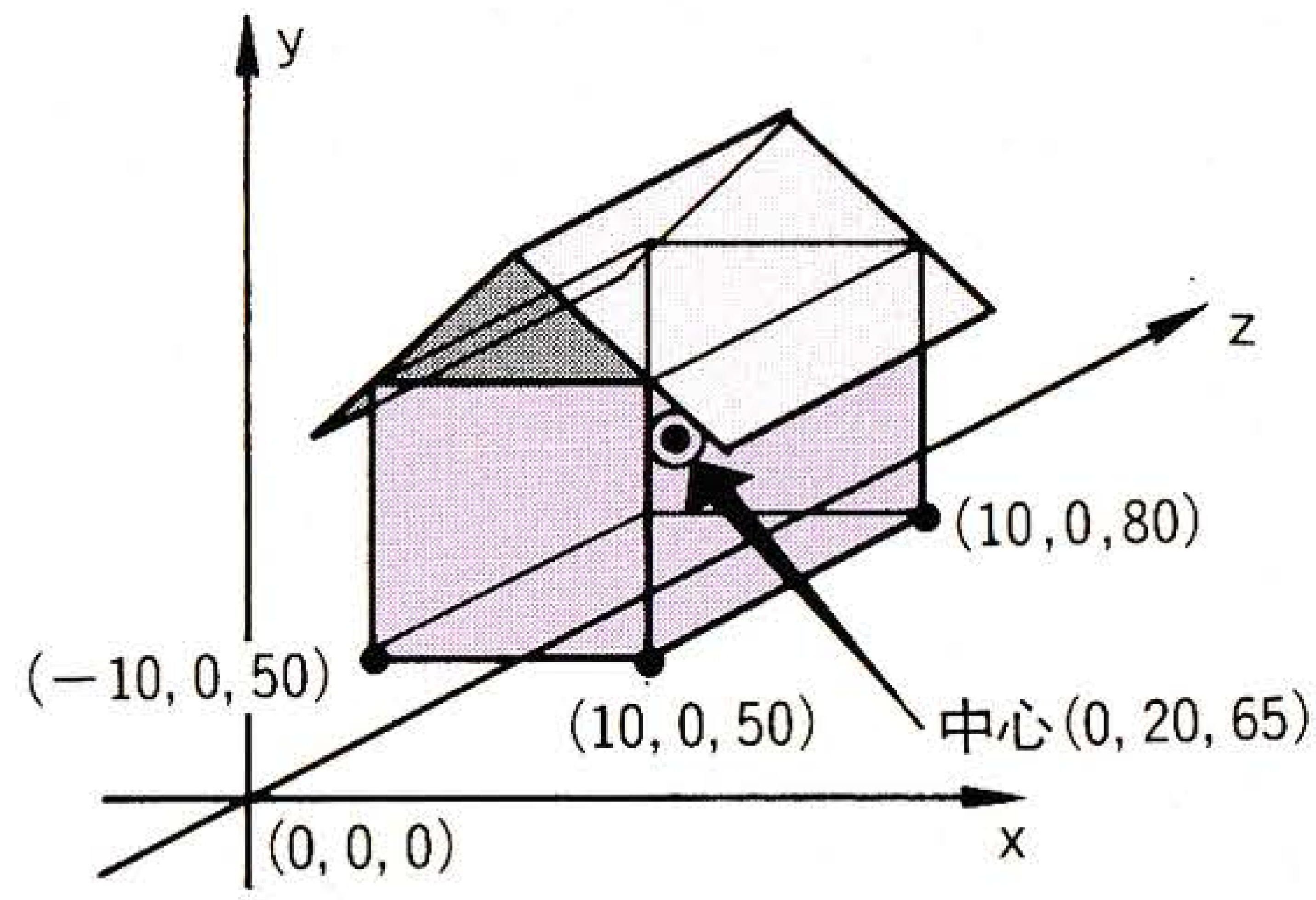


Figure 2

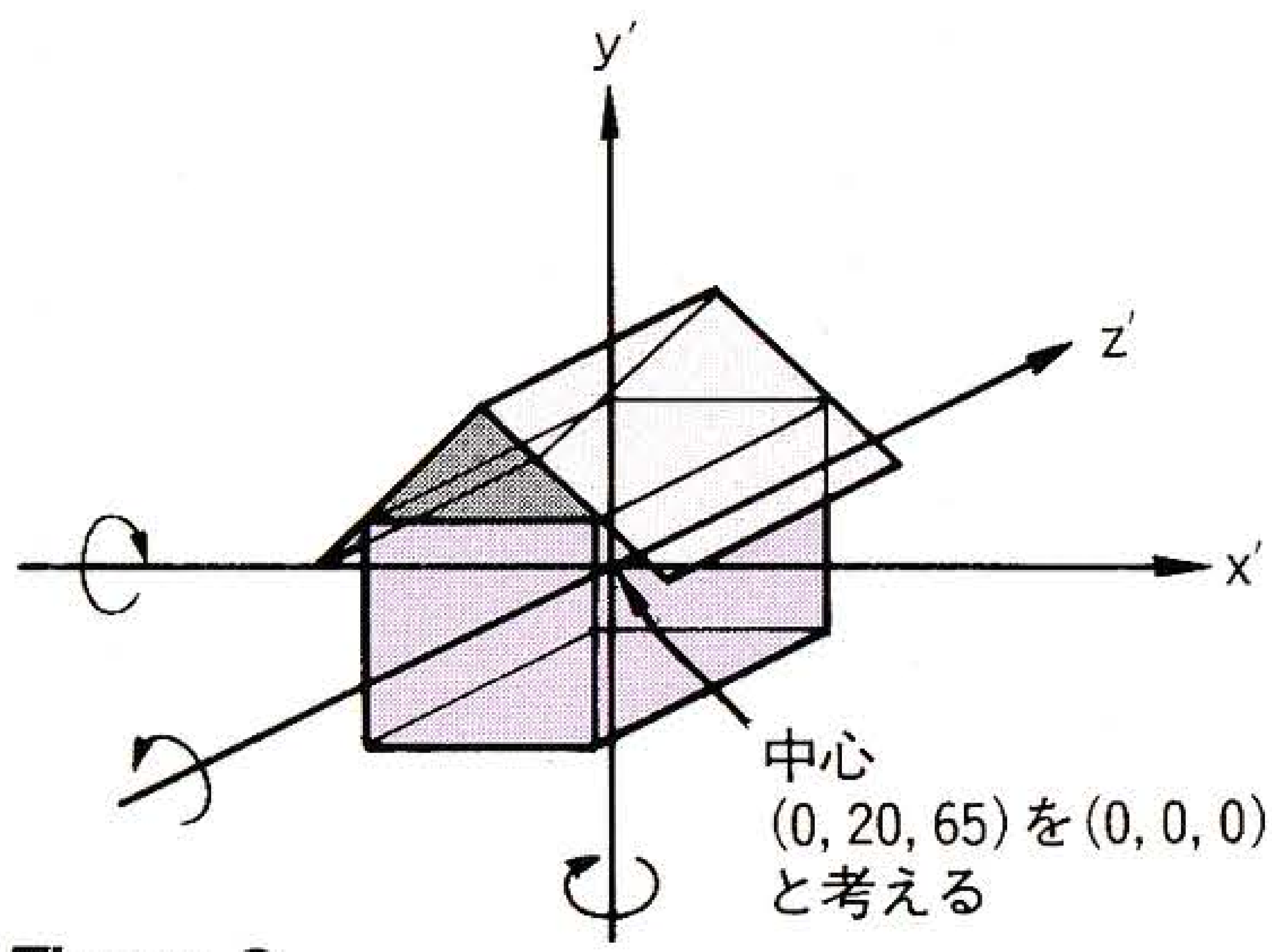


Figure 3

的に直線で表し，その直線の始点，終点を表す座標（x，y，z）を，図形データとします．この図中に紫色の矢印は，自分が見ている方向にあたります．この矢印は，常に固定されているものと考えます．したがって図形をいろいろな方向，距離からながめているように見せるには，この矢印の方向を基準にして図形データの方を回転したり，移動したりしなければならないわけです．

プログラムを実行すると，まず回転する角度を，x 軸回りから順に y 軸， z 軸と聞いてきます．プログラム中に DATA として挿入されている図形の中心となる点（普通は重心）を中心として回転します．たとえば，家の図形データは *Figure 2* のようなもので中心は（0，20，65）ですから，回転を行うときには，*Figure 3* のような軸を考えてやればよいのです．

角度は *Figure 3* のように各軸に対して左回りとなっています．なお，プログラムでは始めに x 軸回りの回転，次に y 軸回り，z軸回りに回転させたとして計算しています．この順序によって回転後の結果に差が出てくるのですが，詳しく知りたい方は，3D グラフィックの入門書を参照してください．

x，y，z 軸まわりにデータを回転させる角度の入力が終わると，次に平行移動量を入力します．

平行移動は，x，y，z 軸方向に，どれだけ移動させるかということを表します．

画面の視野は，固定されています．この視野に収まるように移動させてください．図形を極端に移動させてしまうと，当然視野から図形が消えてしまい何も見えないということになりかねませんので注意してください．

以上の入力によって，画面には三次元図形が実に立体的に出力されます．何かキーを押すと，再びパラメータの入力に戻りますから，再度異なる回転と移動で三次元グラフィックを楽しんでください．

立体データの作り方

ここで，三次元図形を表示するためのデータの作り方について述べておきます．かんたんな立方体について考えてみることにしましょう．

①頂点を決める．

Figure 4 のように，立方体には8つの

Table 1

頂点No.	座　標	頂点No.	座　標
1	(−10，0，30)	5	(−10，20，30)
2	(10，0，30)	6	(10，20，30)
3	(10，0，50)	7	(10，20，50)
4	(−10，0，50)	8	(−10，20，50)

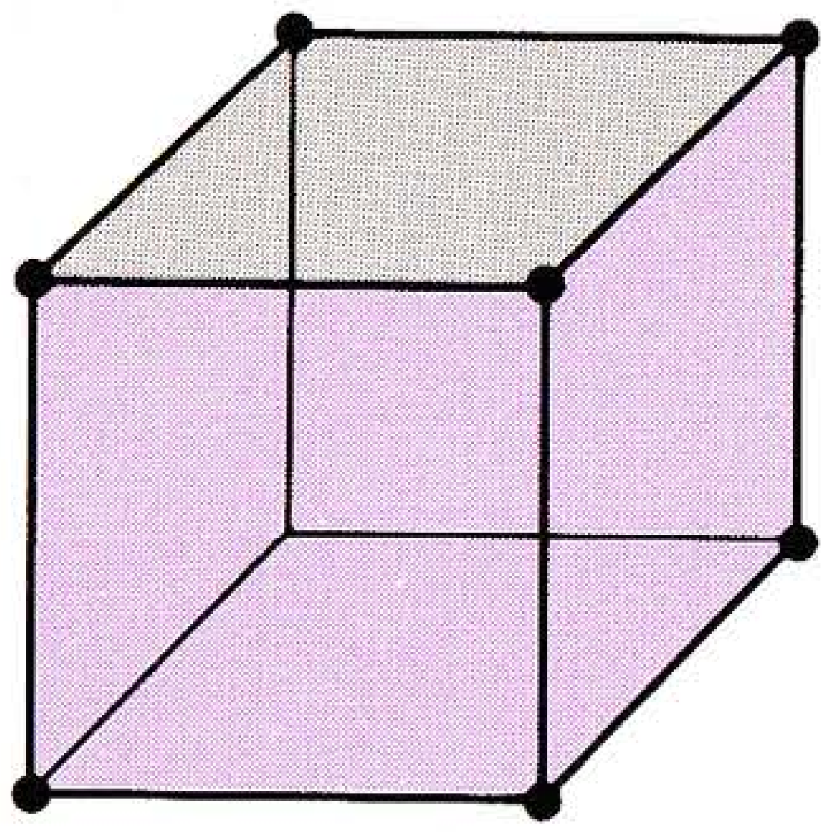


Figure 4

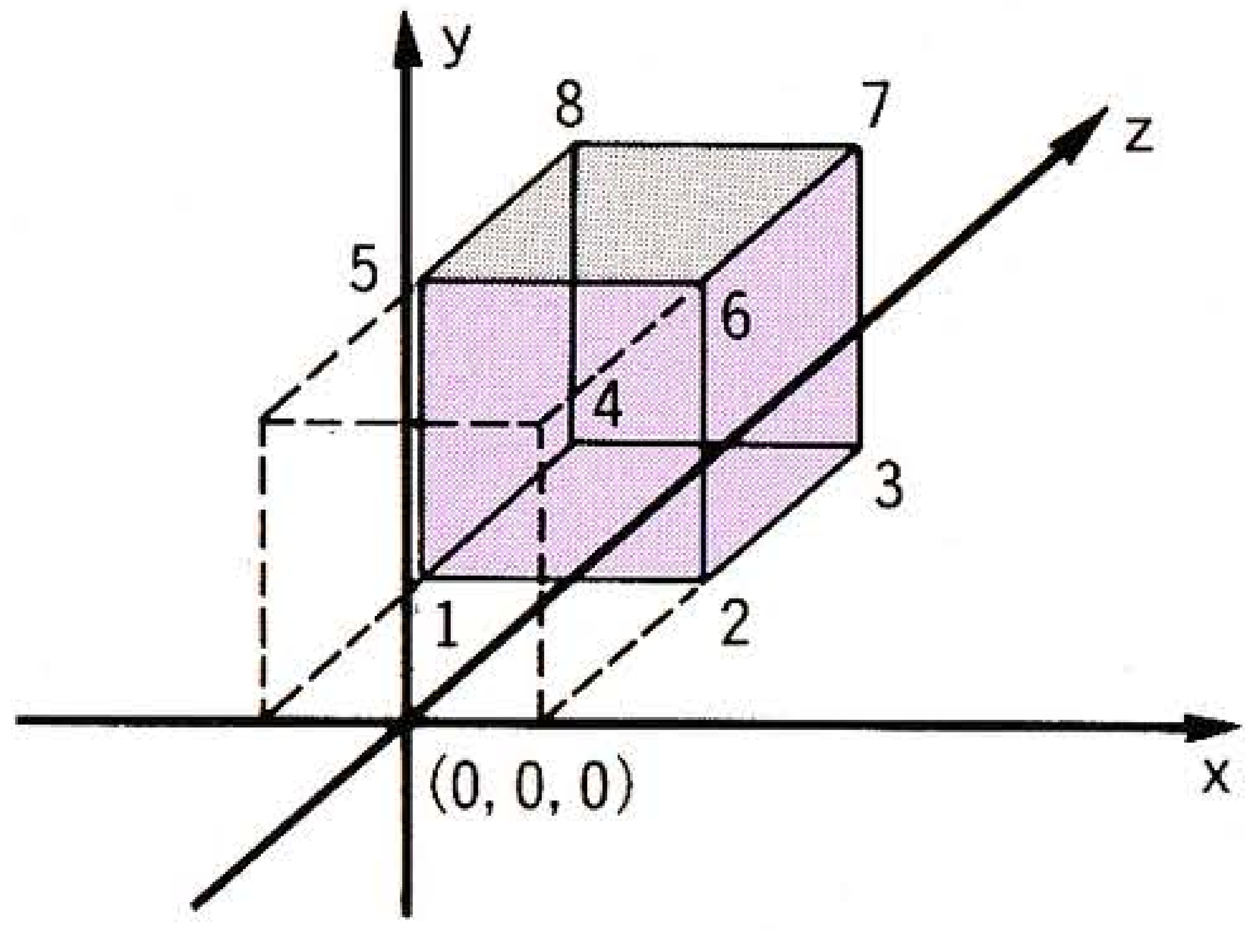


Figure 5

頂点があります．この立方体を，前述のx，y，z座標系の中に置いたとします．それぞれの頂点は，（x，y，z）の3つの数で表現されますが，ここでその頂点の座標を，*Table 1* のようにしてみます（*Figure 5* 参照）．立方体の一辺を20の大きさにとったわけです．この大きさは，あくまでも相対的なものですから，計算の途中で"Over flow error"の出ない大きさであれば，どんなものでもかまいません．

このデータは4020～4050行にDATAとして格納しておきます．またこのときの頂点の個数（立方体なら8）を2020行にDATAの最初の数として格納しておきます．

この図形を回転するときには，立方体の中心（重心）が必要となりますので，3020行にDATAとして格納しておきます．

②直線をどう結ぶのか？

Figure 5 を見てください．直線は頂点2つを結んでいったものですが，結んでほしくない箇所も存在します．そこでデータとして線で結ぶ始点と終点を指示しなくてはなりません．2020行にあるDATAの2番目に結ぶ線の数を格納し，5020行以降に，線で結ぶ始点と終点を格納してください．

以上でデータの登録は終わりました．3Dグラフィックの世界をお楽しみください．

Table 2

2020行	頂点の個数，直線の本数
3020行	x，y，z（中心となる点の座標）
4020～4050行	x_1, y_1, z_1, x_2, y_2, z_2, x_3, y_3, z_3, ………… （3つ一組で頂点の座標を示す）
5020～5040行	b1, e1, b2, e2, b3, e3 ………… （b：直線の始点となる頂点の番号，　e：終点の番号，　これらは，4020行からの頂点の順番による）

List 1

```
100 ' List   1
110 '
120 GOTO 860
130 '
140 ' データの各点を回転及び平行移動させる
150 '
160 GOSUB 370
170 FOR I%=1 TO ND%
180   DX=DX(I%) : DY=DY(I%) : DZ=DZ(I%)
190   GOSUB 480
200   XN(I%)=XN : YN(I%)=YN : ZN(I%)=ZN
210 NEXT I%
220 '
230 ' 直線の視野に入る部分をクリッピングして表示する
240 '
250 SCREEN 2 : COLOR 15,B1,B2 : CLS
260 FOR I%=1 TO NL%
270   LB%=LB%(I%) : LE%=LE%(I%)
280   XB=XN(LB%)  : YB=YN(LB%) : ZB=ZN(LB%)
290   XE=XN(LE%)  : YE=YN(LE%) : ZE=ZN(LE%)
300   GOSUB 610
310   IF FL% THEN GOSUB 750
320 NEXT I%
330 RETURN
340 '
350 ' 回転のマトリクスを計算する
360 '
370  SX=SIN(XR%*DG) : CX=COS(XR%*DG)
380  SY=SIN(YR%*DG) : CY=COS(YR%*DG)
390  SZ=SIN(ZR%*DG) : CZ=COS(ZR%*DG)
400 '
410  PP=CY*CZ : PQ=SX*SY*CZ-CX*SZ : PR=CX*SY*CZ+SX*SZ
420  QP=CY*SZ : QQ=SX*SY*SZ+CX*CZ : QR=CX*SY*SZ-SX*CZ
430  RP=-SY   : RQ=SX*CY          : RR=CX*CY
440  RETURN
450 '
460 ' 点(DX,DY,DZ)が回転して(XN,YN,ZN)となる
470 '
480  DX=DX-HX : DY=DY-HY : DZ=DZ-HZ
490 '
500  XN=DX*PP+DY*PQ+DZ*PR : XN=XN+HX
510  YN=DX*QP+DY*QQ+DZ*QR : YN=YN+HY
520  ZN=DX*RP+DY*RQ+DZ*RR : ZN=ZN+HZ
```

```
530 '
540 '  点(XN,YN,ZN)が,さらに(X,Y,Z)だけ平行移動する
550 '
560  XN=XN+X:YN=YN+Y:ZN=ZN+Z
570  RETURN
580 '
590 '  クリッピングルーチン
600 '
610  C1%=-(XE<-PX*ZE)-2*(XE>PX*ZE)-4*(YE<-PY*ZE)-8*(YE>PY*ZE)
620  C0%=-(XB<-PX*ZB)-2*(XB>PX*ZB)-4*(YB<-PY*ZB)-8*(YB>PY*ZB)
630  IF (C0% OR C1%)=0 THEN FL%=-1  : RETURN
640  IF (C0% AND C1%)<>0 THEN FL%=0 : RETURN
650  IF C0%=0 THEN SWAP XB,XE:SWAP YB,YE:SWAP ZB,ZE:SWAP C0%,C1%
660  IF (C0% AND 1)<>0 THEN T=(PX*ZB+XB)/(PX*(ZB-ZE)+XB-XE):GOTO
 700
670  IF (C0% AND 2)<>0 THEN T=(PX*ZB-XB)/(PX*(ZB-ZE)-XB+XE):GOTO
 700
680  IF (C0% AND 4)<>0 THEN T=(PY*ZB+YB)/(PY*(ZB-ZE)+YB-YE):GOTO
 700
690  T=(PY*ZB-YB)/(PY*(ZB-ZE)-YB+YE)
700  XB=XB+(XE-XB)*T : YB=YB+(YE-YB)*T : ZB=ZB+(ZE-ZB)*T
710  GOTO 620
720 '
730 '  3次元座標→2次元座標の透視変換,及び表示
740 '
750  IF ZB=0 THEN ZB=1
760  IF ZE=0 THEN ZE=1
770  XB=XB/(PX*ZB) : YB=YB/(PY*ZB)
780  XE=XE/(PX*ZE) : YE=YE/(PY*ZE)
790  XB=XB*KX+X0   : YB=Y0-YB*KY
800  XE=XE*KX+X0   : YE=Y0-YE*KY
810  LINE (XB,YB)-(XE,YE),LC%
820  RETURN
830 '
840 '  データの読み込み
850 '
860 RESTORE 2020 : READ ND%,NL%
870 DIM DX(ND%),DY(ND%),DZ(ND%),XN(ND%),YN(ND%),ZN(ND%),LB%(NL%)
,LE%(NL%)
880 '
890 RESTORE 3020 : READ HX,HY,HZ
900 RESTORE 4020
910 FOR I%=1 TO ND% : READ DX(I%),DY(I%),DZ(I%) : NEXT I%
920 RESTORE 5020
930 FOR I%=1 TO NL% : READ LB%(I%),LE%(I%) : NEXT I%
```

```
940 '
950 ' メインルーチン
960 '
970 DG=3.14159/180
980 XS=0 : YS=0 : XM=255 : YM=191
990 X0=(XS+XM)/2 : Y0=(YS+YM)/2
1000 KX=(XM-XS)/2 : KY=(YM-YS)/2
1010 PX=1 : PY=KY/KX
1030 LC%=15 : B1=1 : B2=13
1040 '
1050 SCREEN 1 : COLOR 15,13,13 : WIDTH 29 : KEY OFF
1060 '
1070 PRINT " *** 3D Graphic Package ***" : PLAY "T128VL805A"
1080 PRINT : PRINT "X,Y,Z ソレゾレノ ジクノ センタンノ ホウカラ"
1090 PRINT : PRINT "ミテ、 トケイマワリニ ブッタイヲ カイテンサセルノヲ"
1100 PRINT : PRINT "(+)ノ ホウコウト シマス。"
1110 PRINT : PRINT
1120 INPUT "X ジクノ カイテンカクド ";XR%
1130 INPUT "Y ジクノ カイテンカクド ";YR%
1140 INPUT "Z ジクノ カイテンカクド ";ZR%
1150 '
1160 X=0 : Y=0 : Z=0
1170 PLAY "O5L8A"
1180 PRINT : PRINT "ブッタイノ イドウモ オコナイマスカ ? (y/n)";
1190 I$=INPUT$(1) : PRINT I$
1200 IF I$="N" OR I$="n" THEN 1280
1210 PRINT : PRINT "X,Y,Z ソレゾレノ ジクニ ソッタ ホウコウヲ"
1220 PRINT : PRINT "(+)ト シマス。"
1230 PRINT : PRINT
1240 INPUT "X ジク ホウコウノ イドウ ";X
1250 INPUT "Y ジク ホウコウノ イドウ ";Y
1260 INPUT "Z ジク ホウコウノ イドウ ";Z
1270 '
1280 PRINT : PRINT "ナニカ キーイン シテクダサイ。"
1290 PRINT : PRINT "カイテン・イドウシタ ズケイガ デマス。"
1300 I$=INPUT$(1) : PLAY "L4O3A"
1310 PRINT : PRINT "Ok ! スコシ マッテネ..."
1320 GOSUB 160 : PLAY "L8O5A"
1330 I$=INPUT$(1)
1340 SCREEN 1 : COLOR 15,13,13 : CLS
1350 PRINT "ツヅケマスカ ? (y/n) ";
1360 I$=INPUT$(1) : PRINT I$
1370 IF I$="n" OR I$="N" THEN END
```

```
1380 PRINT
1390 PRINT "ｲﾏﾉ ｶｲﾃﾝ(R)ﾄ ｲﾄﾞｳﾊ､"
1400 PRINT
1410 PRINT "X(R)=";XR%
1420 PRINT "Y(R)=";YR%
1430 PRINT "Z(R)=";ZR%
1440 PRINT
1450 PRINT "X=";X
1460 PRINT "Y=";Y
1470 PRINT "Z=";Z;"  ﾃﾞｼﾀ｡"
1480 PRINT
1490 GOTO 1120
1500 '
2000 ' 点の個数と線の本数
2010 '
2020 DATA 14,17
2030 '
3000 ' 回転の中心となる座標
3010 '
3020 DATA 0,20,65
3030 '
4000 ' 各点の座標
4010 '
4020 DATA 10,0,50, 10,0,80, -10,0,80, -10,0,50
4030 DATA 10,20,50, 10,20,80, -10,20,80, -10,20,50
4040 DATA 15,20,50, 15,20,80, -15,20,80, -15,20,50
4050 DATA 0,35,50, 0,35,80
4060 '
5000 ' 線で結ぶ始点,終点の番号
5010 '
5020 DATA 1,2, 2,3, 3,4, 4,1, 1,5, 2,6, 3,7, 4,8
5030 DATA 9,10, 10,11, 11,12, 12,9
5040 DATA 9,13, 10,14, 11, 14, 12,13, 13,14
```

スプライトパターン ジェネレータ

MSX には，図形を描いたり移動させたりするときに大変便利な，スプライト機能があります．しかしいざ自分でスプライト機能を利用するとなると，スプライトパターン・データを作るのが，意外とやっかいなものだと痛感するでしょう．そこで，画面上でカーソルを移動し，できたパターンをデータとして表示させることにしました．

利用法

プログラムを実行すると，まずスプライトパターンの大きさを聞いてきます．ここで，8×8または16×16のどちらかを選んで入力してください．次に画面にはカラーチャートが表示されます．そして，*Figure 1* のように使う色の数を聞いてきます．

スプライトパターンは単色でしか使えません．もしいろいろな色を使いたいならば，各色ごとにスプライトパターンを作ることになります．この処理をデータ作製時にやってしまおうというわけです．ただし，4色までに留めておいてください．

色の数を入力すると，*Figure 2* のように使う色を聞いてきますので，上に表示されているカラーチャートを見て色に対応する数字を入力してください．このとき，[BS]，[INS]，[DEL] キーなどは使えません．キーを押しまちがえたときは，15より大きい数を入力するとやり直しができます．この操作を使う色数だけ繰り返します．

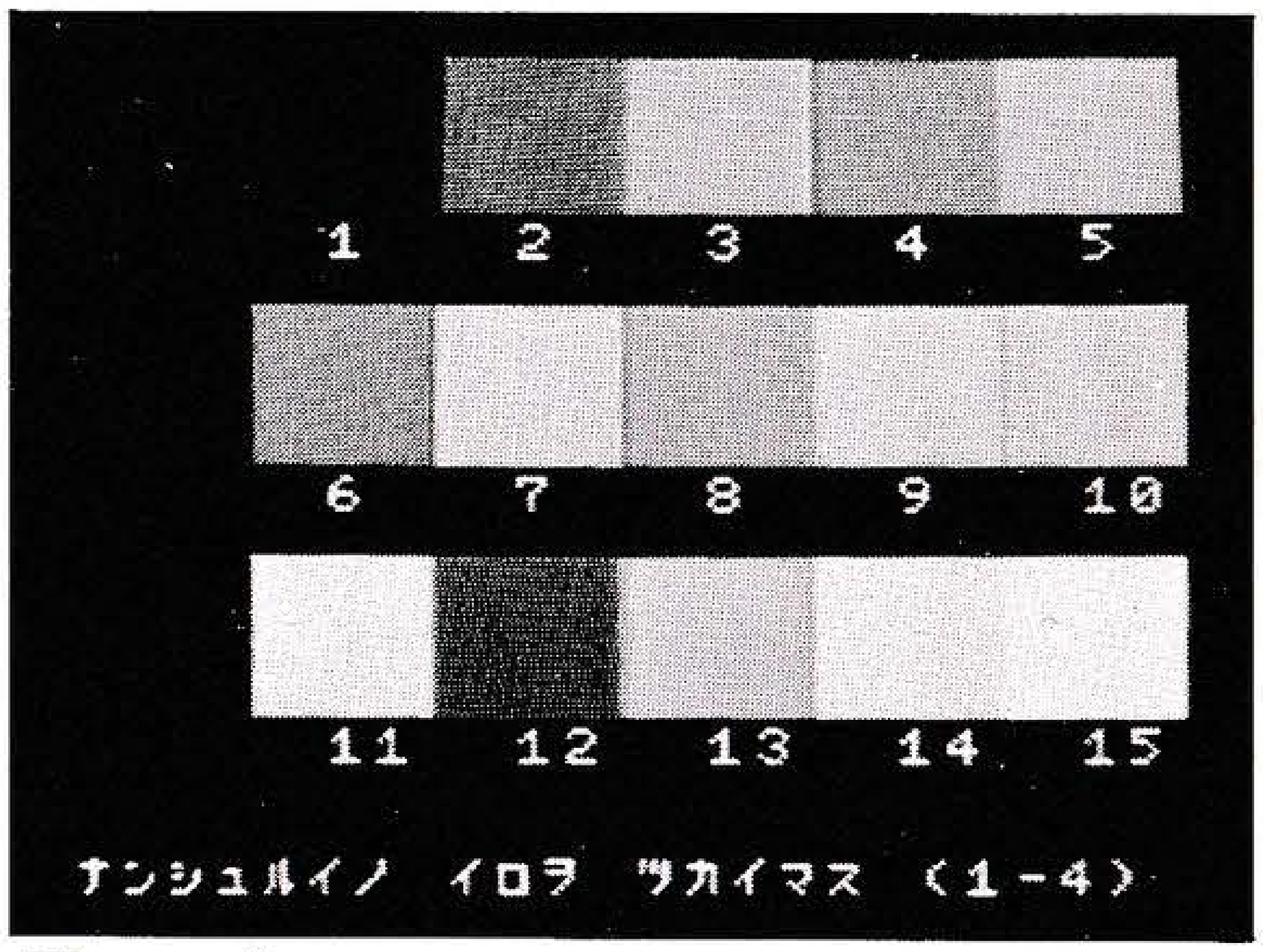


Figure 1

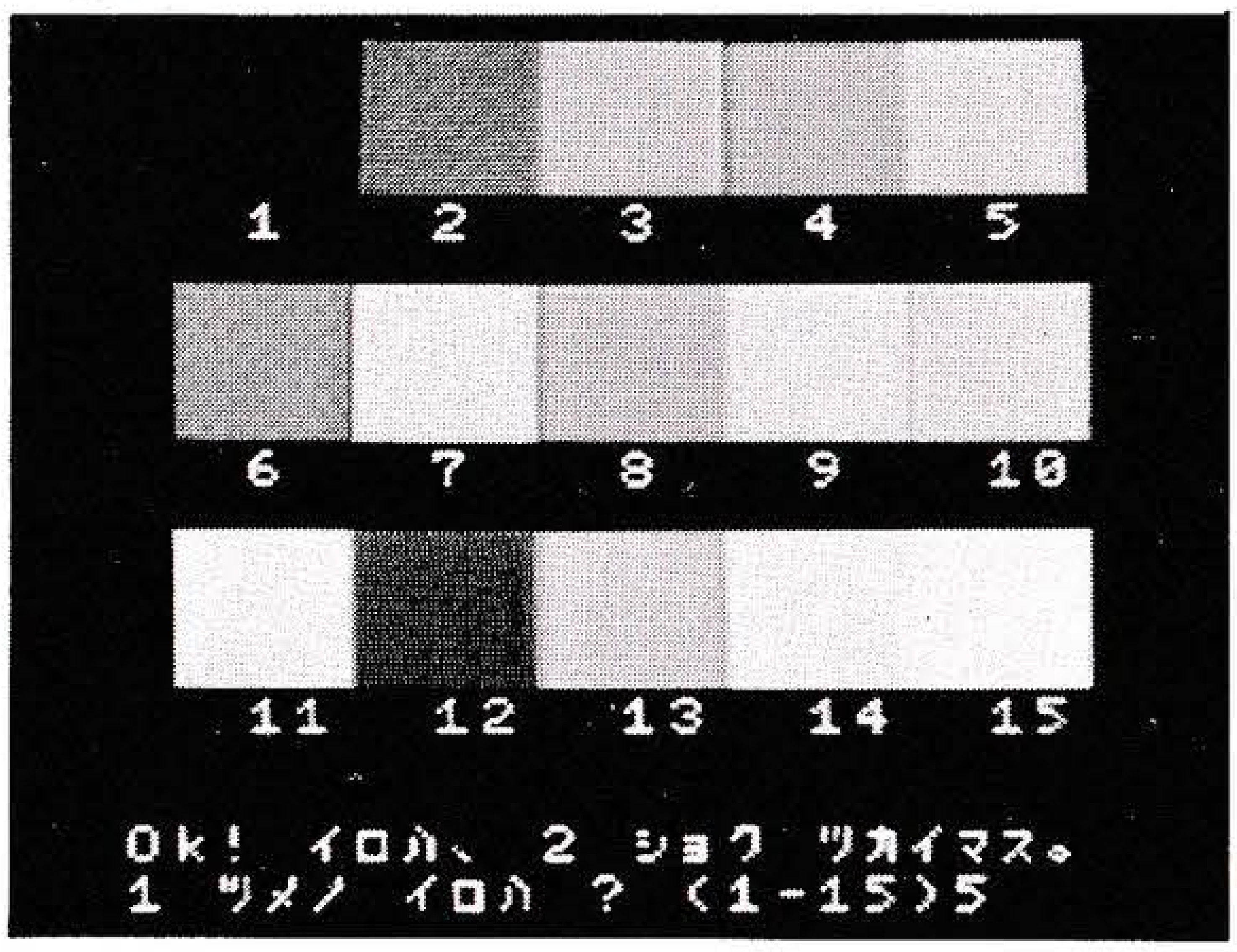


Figure 2

いよいよ図形の定義ですが，*Figure 3* を見てください．

＋印の四角で囲まれたエリアが，スプライトパターンの大きさだと考えてください．中央にあるカーソルを動かし，画面左下に

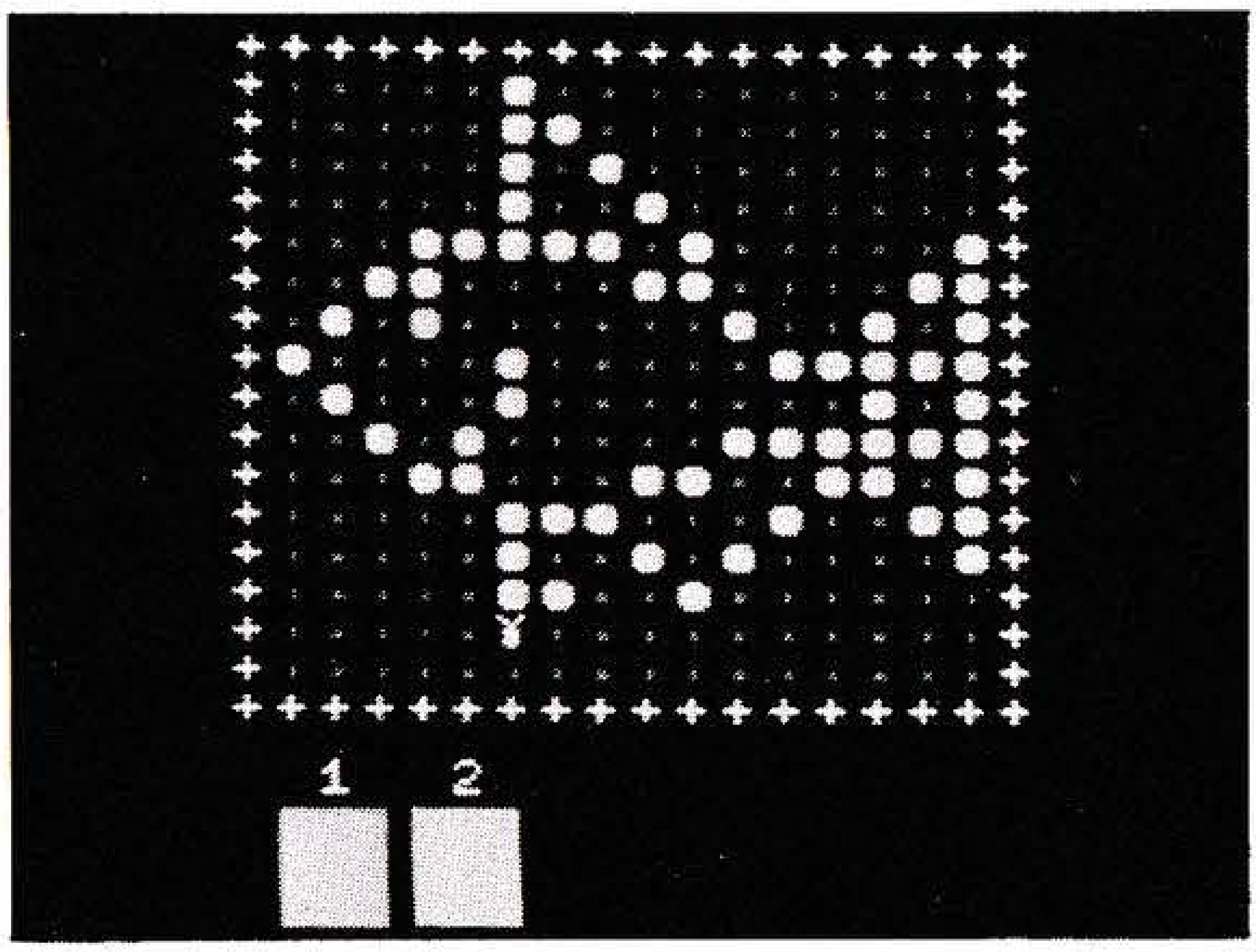


Figure 3

表示されているカラーチャート番号を押すことによって図形を作製します．あるドットを消去したいときには，0キーを押してください．図形が完成したら，RETURN キーを押します．

RETURN キーを押すと，

```
Color = n
  0,124,  5,・・・・・
```

とデータが表示されますから，ノートに書き写します．このデータが，いわゆるスプライトパターン・データなのですが，これは，スプライトパターンサイズが8×8の場合と16×16の場合で若干異なります．

●8×8のとき

COLOR ＝ n のあとに8つのデータがまず出力されましたね．これが，色をn番として表示すべきスプライトパターン・データです．2色以上使っている場合は，何かキーを押すことで，次々と，その色について8個ずつのデータが表示されます．

●16×16のとき

1つの COLOR について，32個表示されます．16×16のサイズでは，1つの図形を表すのに32個のデータが必要なのです．

さて，ここで出力された値をどうやって使うのか，以下に例を示しながら説明しましょう．

8×8のサイズで次のようにデータが出力されたとします．

```
Color = 7
 60,126,195,195,195,195,126, 60
Hit any key...
Color = 13
  0,  0, 60, 60, 60, 60,  0,  0
Hit any key...
--- END ---
```

このパターンは，7と13の2色を使ったことがわかりますね．これを SCREEN 2 の座標（100，100）に表示してみることにします．プログラムは*List 2a* のようになります．

16×16のサイズもスプライトパターン・データを定義するときに，それぞれデータが32個いること，スクリーンモードを設定するとき，SCREEN 2, 2としなくてはならないことの2つを除けば 8×8 の場合と同様です．

List 2

```
100 ' List   2
110 '
120 DEFSNG A-Z : PLAY "T120V12L32"
130 '
140 ' 使うスプライトパターンの大きさ,数(色)の設定
150 '
160 SCREEN 0,,0 : COLOR 15,4,4 : WIDTH 38 : KEY OFF : CLS
170 LOCATE 7,10 : PRINT "*** Sprite Editor ***" : PLAY "O6A"
180 LOCATE 2,12 : PRINT "ﾄﾞﾁﾗﾉ ｻｲｽﾞ ? (1) 8*8, (2) 16*16 ";
190 I$=INPUT$(1) : IF I$<"1" OR I$>"2" THEN PLAY "O3AA" : GOTO 1
90
200 PRINT I$ : SZ=VAL(I$) : XW=8*SZ : YW=8*SZ
210 '
220 SCREEN 2,2 : COLOR 15,0,0 : CLS
230 OPEN "GRP:" FOR OUTPUT AS #1
240 FOR I=1 TO 15
250   X=((I-1) MOD 5)*32+48 : Y=((I-1) ¥ 5)*50
260   LINE (X,Y)-(X+31,Y+31),I,BF
270   X=X+6 : Y=Y+34 : DRAW "BM=X;,=Y;" : PRINT #1,I
280 NEXT I
290 DRAW "BM18,160" : PRINT #1,"ﾅﾝｼｭﾙｲﾉ ｲﾛｦ ﾂｶｲﾏｽ (1-4) " : PLAY
 "O6A"
300 I$=INPUT$(1) : IF I$<"1" OR I$>"4" THEN PLAY "O3AA" : GOTO 3
00
310 CN=VAL(I$)
320 LINE (0,160)-(255,167),1,BF
330 DRAW "BM40,160" : PRINT #1,"Ok! ｲﾛﾊ､";CN;"ｼｮｸ ﾂｶｲﾏｽ｡"
340 FOR I=1 TO CN
350   LINE (0,170)-(255,177),1,BF : IN$=""
360   DRAW "BM32,170" : PRINT #1,I;"ﾂﾒﾉ ｲﾛﾊ ? (1-15) " : PLAY "O
6A"
370   I$=INPUT$(1) : PLAY "O4A" : IF I$=CHR$(13) THEN 400
380     IN$=IN$+I$ : DRAW "BM184,170" : PRINT #1,IN$
390   GOTO 370
400   C=VAL(IN$): IF C<1 OR C>15 THEN PLAY "O3AA" : GOTO 350
410   C(I)=C
420 NEXT I
430 PLAY "O4AAR16O3AA" : COLOR 15 : CLS
440 '
450 DIM D(XW,YW)
460 DRAW "S4A0 BM0,0"
470 PRINT #1,"  ";STRING$(XW+2,"+")
```

```
480 FOR I=1 TO YW
490   COLOR 15 : PRINT #1,"  +";
500   COLOR 4  : PRINT #1,STRING$(XW,"·");
510   COLOR 15 : PRINT #1,"+"
520 NEXT I
530 PRINT #1,"  ";STRING$(XW+2,"+")
540 '
550 FOR I=1 TO CN
560   X=I*24
570   LINE (X,160)-(X+19,183),C(I),BF
580   DRAW "BM=X;,150" : PRINT #1,I
590 NEXT I
600 '
610 ' スプライトパターンの作製
620 '
630 FOR I=1 TO 8 : READ X(I),Y(I) : NEXT I
640 DATA 0,-1, 1,-1, 1,0, 1,1, 0,1, -1,1, -1,0, -1,-1
650 X=XW/2 : Y=YW/2
660 '
670 XG=(X+2)*8 : YG=Y*8
680 COLOR 15 : DRAW "BM=XG;,=YG;" : PRINT #1,"¥"
690 PLAY "O4A"
700 '
710 I=STICK(0) : IF I=0 THEN 790
720   XS=X(I) : YS=Y(I)
730   IF X+XS<1 OR Y+YS<1 OR X+XS>XW OR Y+YS>YW THEN 790
740   C=D(X,Y) : IF C=0 THEN CH$="·" : C=4 ELSE CH$="●"
750   LINE (XG,YG)-(XG+7,YG+7),1,BF
760   COLOR C : DRAW "BM=XG;,=YG;" : PRINT #1,CH$
770 X=X+XS : Y=Y+YS : GOTO 670
780 '
790 I$=INPUT$(1)
800 IF I$=CHR$(13) THEN 880
810 IF I$="0" THEN D(X,Y)=0 : XS=0 : YS=0 : GOTO 740
820 '
830 I=VAL(I$) : IF I<1 OR I>CN THEN 710
840 D(X,Y)=C(I) :  XS=0 : YS=0 : GOTO 740
850 '
860 ' スプライトパターン・データの表示
870 '
880 SCREEN 0 : COLOR 15,4,4 : WIDTH 38 : CLS
890 LOCATE 7,0 : PRINT "*** Sprite Pattern **" : PRINT : PRINT
```

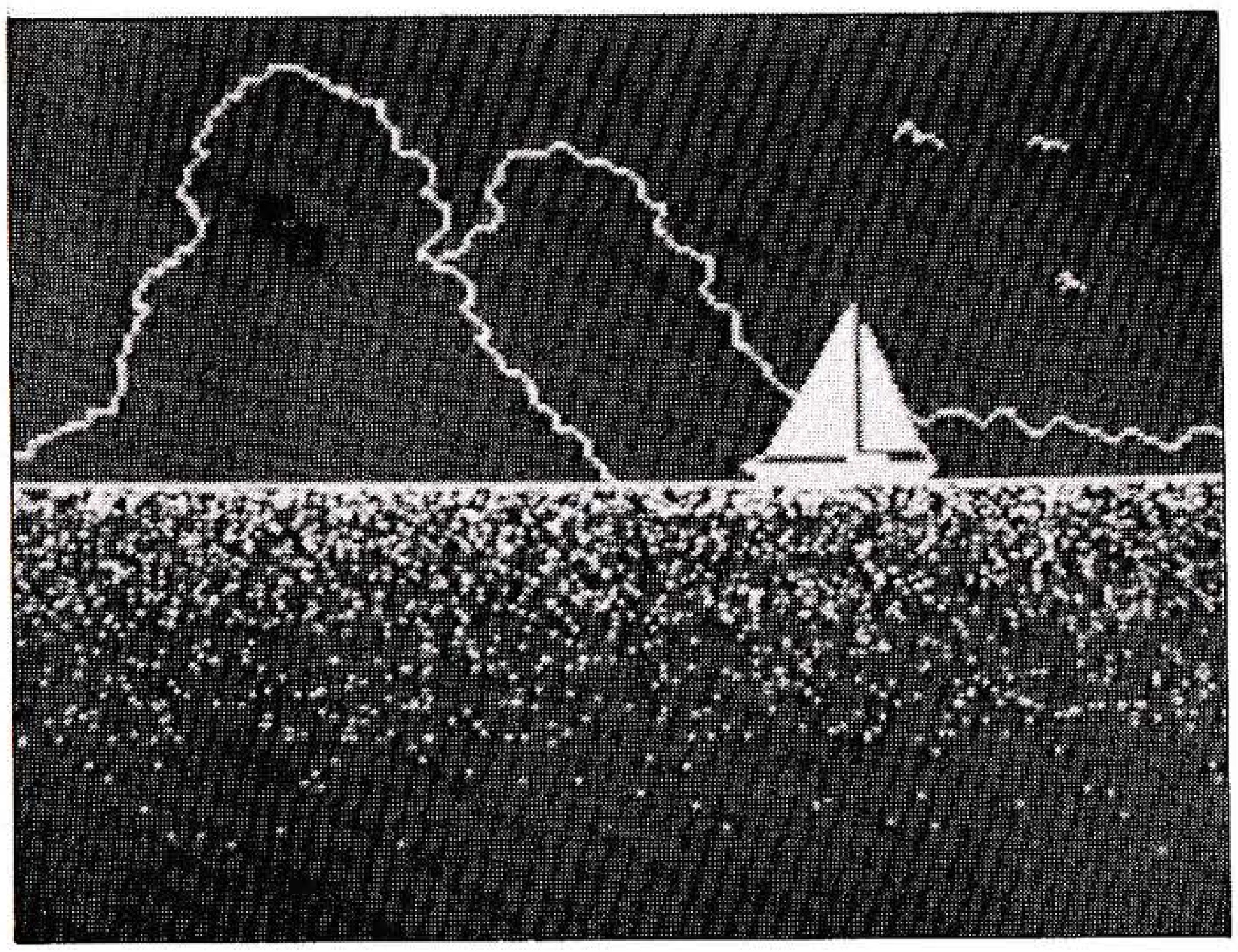
```
900 FOR I=1 TO CN
910   PRINT "Color =";C(I) : PRINT
920   FOR XD=0 TO (SZ-1)*8 STEP 8
930     FOR YD=0 TO (SZ-1)*8 STEP 8
940       FOR Y=1 TO 8
950         D$=""
960         FOR X=1 TO 8
970           IF D(XD+X,YD+Y)=C(I) THEN D$=D$+"1" ELSE D$=D$+"0"
980         NEXT X
990         PRINT USING "###";VAL("&B"+D$);
1000         IF Y<8 THEN PRINT ",";
1010       NEXT Y
1020       PRINT
1030     NEXT YD
1040   NEXT XD
1050   PRINT : PRINT "Hit any key..."; : DM$=INPUT$(1)
1060   PRINT : PRINT
1070 NEXT I
1080 PRINT "--- END ---":END
```

List 2a

```
100 ' List   2a
110 '
120 '
130 A$=CHR$(60)+CHR$(126)+STRING$(4,CHR$(195))+CHR$(126)+CHR$(6
0)
140 B$=CHR$(0)+CHR$(0)+STRING$(4,CHR$(60))+CHR$(0)+CHR$(0)
150 SCREEN 2,0
160 SPRITE$(0)=A$ : SPRITE$(1)=B$
170 PUT SPRITE 0,(100,100),7,0
180 PUT SPRITE 1,(100,100),13,1
190 I$=INPUT$(1) : END
```

MSX グラフィックエディタ

私たちが幼い頃から慣れ親しんできた絵には，クレヨンや色エンピツで描いたものもあれば，水彩絵の具，油絵の具など様々なもので描かれています．これらは，それぞれ独特の雰囲気を持っているので，好みが分かれるところでしょう．このMSX・グラフィックエディタは，コンピュータで絵を描くという，新しいタイプの画材だと思ってください．これも，水彩や油彩などとは別の，新しい感覚の絵を作り出してくれることでしょう．あとは，あなたの感性にまかせて，筆をすすめていくだけです．さあ，さっそくこのグラフィックエディタで，絵を描いてみてください．



利用法

このプログラムは，32K以上のRAM容量がないと動きません．したがってもし，あなたのMSXが16K RAMの記憶容量しか持たないとしたら，後述の16K RAMへの対応の説明をみて，プログラムを若干変更しなければなりません．

このツールは，3種類の太さを持った筆と，3種類の太さに描けるブラシを使って絵を描いていくものです．

■カーソルについて

カーソルは2つあります．1つは数字などが一緒になっているもの，もう一つはただの十字のものです．以後，前者をメインカーソル，後者をLPカーソルと呼びます．

すべての描写は，メインカーソルを中心に行います．このメインカーソルは，2つの情報を持っています．1つは，カーソル左上の文字，これは，キー[B]を押すことでBに，[L]でLに変化します．このBとLはそれぞれ，ブラシモードと，ラインモードを示しています．ブラシモードでは，あたかもスプレーを吹きつけた様な効果が得られます．これに対して，ラインモードでは，筆で描いたような効果が得られます．これら筆やブラシの太さが，右下の数字で

表されています.

LP カーソルは，直線や四角形，円を描くときに利用します.

■ キー操作

ここで，すべてのキー操作を説明します.

[B]：ブラシモードの選択.

[L]：ラインモードの選択.

[カーソルキー]：メインカーソルの移動.

[カーソルキー + SHIFT]：メインカーソルを10ドット飛びに移動させる.

[1]，[2]，[3]：筆またはブラシの太さを選ぶ

[スペースバー]：メインカーソルの位置に，そのカーソルの情報に従って，点もしくはブラシ状の点を描く.

スペースバーを押しながらカーソルキーを押すと，メインカーソルは軌跡を残しながら移動します.

[RETURN]：メインカーソルの位置に LP カーソルを移動させる.

[RETURN + SHIFT]：LP カーソルの位置にメインカーソルを移動させる.

[F 1]：メインカーソルと LP カーソルの間に直線を引く．その後，メインカーソルの位置に LP カーソルを移動させる.

[F 6]：[F 1] と同様に直線を引くが，LP カーソルを移動させない.

[F 2]：メインカーソルと LP カーソルを対角線とする四角形を描く．描いている途中，[ESC] キーで中止することができる.

[F 7]：メインカーソルと LP カーソルを対角線とする四角形を塗りつぶす．ブラシモード（メインカーソル左上が B）のときは，四角形の内部にランダムに点を打つ．好みの濃さになったら，[ESC] キーを押して中断する.

[F 3]：LP カーソルを中心として，メインカーソル上に円を描く．途中，[ESC] キーで中止することができる.

[F 8]：LP カーソルを中心としてメインカーソルを通る円をモアレ状にぬりつぶす．ブラシモードでは，その円の内部にランダムに点を打つので，適当な密度になったら，[ESC] キーを押して中断する.

[F 4]：メインカーソルを中心にして，ペイントを行う.

[F 5]：メインカーソルと LP カーソルを結んでバックグラウンドカラーで直線を引く．途中，カーソルキーでメインカーソルを移動することができその間，メインカーソルと LP カーソルで結ばれた直線の通った部分は消去されることになる．[ESC] キーで中断することができる.

[SELECT]：バックグラウンドカラーとフォアグラウンドカラーを入れかえる.

例えば，直前に描いた円を消したいとき，このキーを押して，もう一度同じ円を描く操作をすれば，その円はバックグラウンドカラーの円で重ね書きされたことになり，消えてしまう．再びこのキーを押して，色をもとにもどすことができる.

[DEL]：終了する.

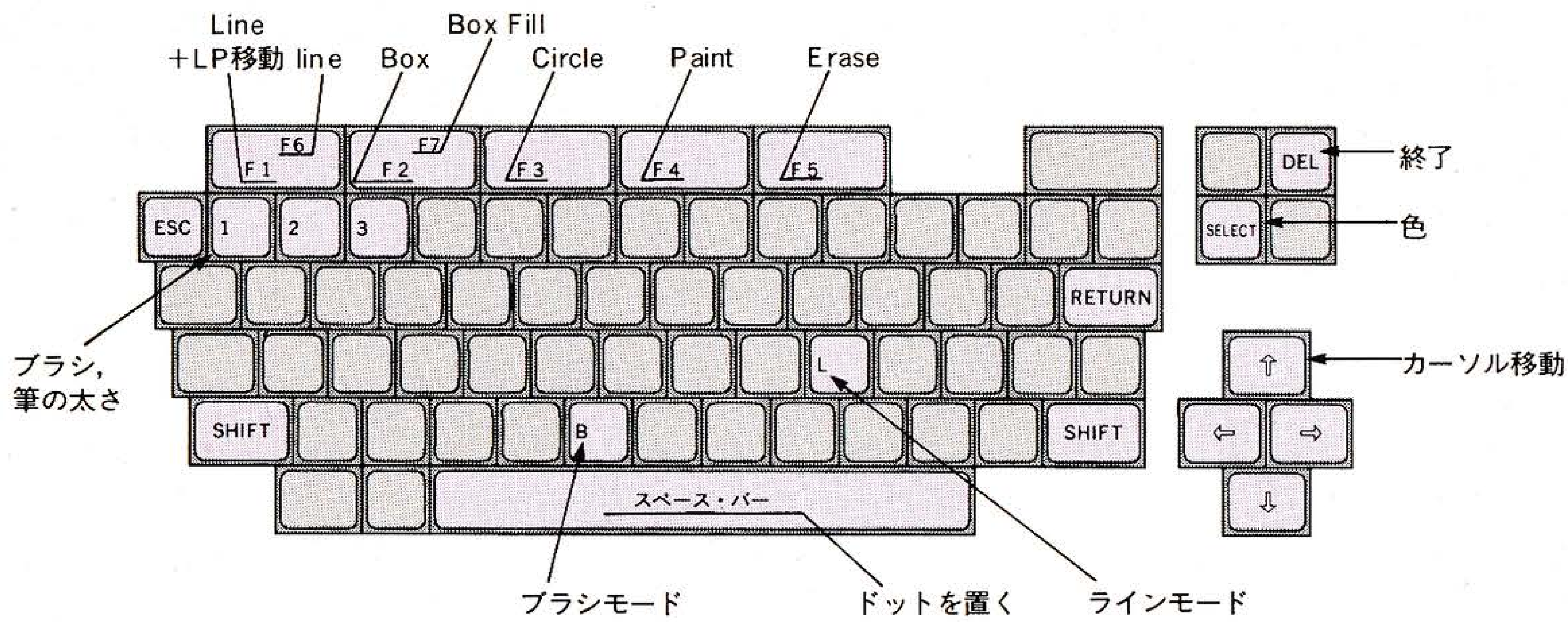


Figure 1

では，プログラムを実行してください．

```
Load file name=
```

のように表示されます．ここではもし描きかけの絵があったらそのファイル名を入力する．なければ RETURN キーを押してください．このように表示されます．

```
Save file name=
```

ここでは描いた絵をセーブするときのファイル名を入力します．グラフィックエディタを終了するとき，DEL キーを押しますが，このときカセットテープレコーダも録音状態にしておくこと．データをセーブするつもりのないときは，RETURN キーを押す．

入力後に，もしロードするファイルのある場合は，カセットテープレコーダを再生状態にしておくと，データは自動的に読み込まれ，その画面が再現されます．

以上が，キー操作の説明でした．どうも複雑でよくわからないという人のために，*Figure 1* を見ながら1度操作してみてください．

あとは，アキルまでこのグラフィックエディタを使えば良いだけです．

List 3

```
100 ' List  3
110 '
120 CLEAR 300,&HB9FF
130 DEFSNG A-Z
140 DA=&HBA00
```

```
150 '
160 PI=3.14159 : DG=PI/180
170 DV$="CAS:"
180 FC=15 : BC=4
190 GOSUB 3490
200 '
210 K1$=CHR$(223)+CHR$(191)+CHR$(127)+CHR$(222)+CHR$(190)+CHR$(1
26)
```

キー入力判定用文字列

```
220 K2$=CHR$(254)+CHR$(253)+CHR$(191)
```

キー入力判定用文字列

```
230 '
240 BR(1)=1 : BR(2)=2 : BR(3)=3
250           LR(2)=1 : LR(3)=2
260 RF=1    : MF$="L"
270 '
280 CX=128  : CY=91   : LX=128 : LY=91
290 CC=8    : CL=8    : CM=CC
300 '
310 FOR I=1 TO 10
320   KEY I,""
330 NEXT I
340 '
350 RESTORE 390
360 FOR I=1 TO 8
370   READ XS(I),YS(I)
380 NEXT I
390 DATA 0,-1, 1,-1, 1,0, 1,1
400 DATA 0,1, -1,1, -1,0, -1,-1
410 '
420 RESTORE 460 : CL$=""
430 FOR I=1 TO 32
440   READ D : CL$=CL$+CHR$(D)
450 NEXT I
460 DATA 65,65,65,65,65,121,0,252
470 DATA 0,1,1,1,1,1,1,0
480 DATA 0,0,0,0,0,0,0,126
490 DATA 0,0,0,0,0,0,0,0
```

スプライト設定用の文字列定義“L”のカーソルパターン

```
500 '
510 RESTORE 550 : CB$=""
520 FOR I=1 TO 32
530   READ D : CB$=CB$+CHR$(D)
540 NEXT I
550 DATA 113,73,113,73,73,113,0,252
560 DATA 0,1,1,1,1,1,1,0
570 DATA 0,0,0,0,0,0,0,126
580 DATA 0,0,0,0,0,0,0,0
```

“B”のカーソルパターン

```
590
600 RESTORE 640 : CP$=""
610 FOR I=1 TO 32
620   READ D : CP$=CP$+CHR$(D)
630 NEXT I
640 DATA 0,0,1,1,1,1,3,62
650 DATA 3,1,1,1,1,0,0,0
660 DATA 0,0,0,0,0,0,128,248
670 DATA 128,0,0,0,0,0,0,0
```

十字形のカーソルパターン

```
680 '
690 RESTORE 730 : C$(1)=""
700 FOR I=1 TO 32
710   READ D : C$(1)=C$(1)+CHR$(D)
720 NEXT I
730 DATA 0,0,0,0,0,0,0,0
740 DATA 0,0,0,0,0,0,0,0
750 DATA 0,0,0,0,0,0,0,0
760 DATA 0,8,24,8,8,8,28,0
```

"1"のパターン

```
770 '
780 RESTORE 820 : C$(2)=""
790 FOR I=1 TO 32
800   READ D : C$(2)=C$(2)+CHR$(D)
810 NEXT I
820 DATA 0,0,0,0,0,0,0,0
830 DATA 0,0,0,0,0,0,0,0
840 DATA 0,0,0,0,0,0,0,0
850 DATA 0,24,36,8,16,32,60,0
```

"2"のパターン

```
860 '
870 RESTORE 910 : C$(3)=""
880 FOR I=1 TO 32
890   READ D : C$(3)=C$(3)+CHR$(D)
900 NEXT I
910 DATA 0,0,0,0,0,0,0,0
920 DATA 0,0,0,0,0,0,0,0
930 DATA 0,0,0,0,0,0,0,0
940 DATA 0,60,4,24,4,36,24,0
```

"3"のパターン

```
950 '
960 SPRITE$(0)=CL$ : SPRITE$(1)=C$(1) : SPRITE$(2)=CP$
970 '
980 GOSUB 1250 : GOSUB 1310
990 '
1000 GOSUB 1610
1010 I=STRIG(0)
1020 I$=INKEY$
```

```
1030 '
1040 IF I$=CHR$(127) THEN 3760
1050 IF I=-1 THEN GOSUB 1870 : GOTO 1000
1060 IF I$=CHR$(13) THEN GOSUB 1750 : GOTO 1000
1070 IF I$="B" OR I$="b" THEN GOSUB 1980 : GOTO 1000
1080 IF I$="L" OR I$="l" THEN GOSUB 2020 : GOTO 1000
1090 I=VAL(I$)
1100 IF I>0 AND I<4 THEN GOSUB 1920 : GOTO 1000
1110 '
1120 IK=6 : GOSUB 1360
1130 I1=INSTR(K1$,CHR$(IK))
1140 IK=7 : GOSUB 1360
1150 I2=INSTR(K2$,CHR$(IK))
1160 '
1170 IF I1=0 OR I1>6 THEN 1200
1180 ON I1 GOSUB 2080,2400,2670,2100,3040,3210
1190 GOTO 1000
1200 ON I2 GOSUB 2860,2920,3410
1210 GOTO 1000
1220 '
1230 ' cursor
1240 '
1250 PUT SPRITE 0,(CX-7,CY-8),CC,0
1260 PUT SPRITE 1,(CX-7,CY-8),CC,1
1270 RETURN
1280 '
1290 ' LP
1300 '
1310 PUT SPRITE 2,(LX-7,LY-8),CL,2
1320 RETURN
1330 '
1340 ' key
1350 '
1360 OUT 170,(INP(170) AND &HF0) OR IK
1370 IK=INP(169)
1380 RETURN
1390 '
1400 ' dot & brush
1410 '
1420 IF CX<0 OR CX>255 THEN RETURN
1430 IF CY<0 OR CY>191 THEN RETURN
1440 IF MF$="B" THEN 1530
1450 IF RF=1 THEN PSET (CX,CY),FC : RETURN
```

（1180行）左から順に F1 , F2 , F3 , F5 , F6 , F8 のファンクションキーを押した場合の処理

（1200行）左から順に, F4 , F5 , SELECT を押した場合の処理

（1360〜1380行）先行入力なしでキー入力を行うための高級テクニック

```
1460 CIRCLE (CX,CY),LR(RF),FC
1470 PAINT (CX,CY),FC
1480 FOR II=1 TO LR(RF)
1490   CIRCLE (CX,CY),II,FC
1500 NEXT II
1510 RETURN
1520 '
1530 BR=BR(RF)
1540 FOR II=1 TO BR^2*2
1550   RR=INT(RND(1)*(BR+1))
1560   TT=RND(1)*PI*2
1570   PSET (CX+RR*SIN(TT),CY+RR*COS(TT)),FC
1580 NEXT II
1590 RETURN
1600 '
1610 ' cursor
1620 '
1630 I=STICK(0)
1640 IK=6 : GOSUB 1360
1650 IF I=0 THEN RETURN
1660 IF IK=254 THEN K=10 ELSE K=1
1670 IF CX+XS(I)*K<0 OR CX+XS(I)*K>255 THEN RETURN
1680 IF CY+YS(I)*K<0 OR CY+YS(I)*K>191 THEN RETURN
1690 CX=CX+XS(I)*K:CY=CY+YS(I)*K
1700 GOSUB 1250
1710 RETURN
1720 '
1730 ' cursor=LP
1740 '
1750 IK=6 : GOSUB 1360
1760 IF IK=254 THEN 1810
1770 LX=CX : LY=CY
1780 GOSUB 1310
1790 RETURN
1800 '
1810 CX=LX : CY=LY
1820 GOSUB 1250
1830 RETURN
1840 '
1850 ' point set
1860 '
1870 GOSUB 1420
1880 RETURN
```

```
1890 '
1900 ' 1-3
1910 '
1920 RF=I
1930 SPRITE$(1)=C$(RF)
1940 RETURN
1950 '
1960 ' Select L or B
1970 '
1980 MF$="B"
1990 SPRITE$(0)=CB$
2000 RETURN
2010 '
2020 MF$="L"
2030 SPRITE$(0)=CL$
2040 RETURN
2050 '
2060 ' Line
2070 '
2080 F=-1
2090 '
2100 IF RF=1 AND MF$<>"B" THEN LINE (CX,CY)-(LX,LY),FC:GOTO 2340
2110 '
2120 IF CX=LX AND CY=LY THEN RETURN
2130 X1=SQR((CX-LX)^2+(CY-LY)^2)
2140 XS=(CX-LX)/X1:YS=(CY-LY)/X1
2150 C=CC  : CC=1  : GOSUB 1250 : CC=C
2160 C=CL  : CL=1  : GOSUB 1310 : CL=C
2170 X2=CX : Y2=CY : CX=LX : CY=LY
2180 '
2190 IF MF$="L" THEN 2280
2200 R=BR(RF)
2210 FOR I=0 TO X1/R
2220   GOSUB 1420
2230   IF INKEY$=CHR$(27) THEN 2330
2240   CX=CX+XS*R  : CY=CY+YS*R
2250 NEXT I
2260 GOTO 2330
2270 '
2280 FOR I=0 TO X1
2290   GOSUB 1420
2300   IF INKEY$=CHR$(27) THEN 2330
2310   CX=CX+XS : CY=CY+YS
2320 NEXT I
2330 CX=X2 : CY=Y2
2340 IF F=-1 THEN LX=CX : LY=CY : F=0
2350 PLAY "L64O4A" : GOSUB 1250 : GOSUB 1310
2360 RETURN
```

```
2370 '
2380 ' box
2390 '
2400 IF RF=1 AND MF$<>"B" THEN LINE (CX,CY)-(LX,LY),FC,B:RETURN
2410 '
2420 IF CX=LX AND CY=LY THEN RETURN
2430 X3=CX:Y3=CY
2440 IF CX>LX THEN X1=LX : X2=CX ELSE X1=CX : X2=LX
2450 '
2460 C=CC : CC=1 : GOSUB 1250 : CC=C
2470 C=CL : CL=1 : GOSUB 1310 : CL=C
2480 IF MF$="B" THEN R=BR(RF) ELSE R=1
2490 FOR CX=X1 TO X2 STEP R
2500   CY=Y3 : GOSUB 1420
2510   CY=LY : GOSUB 1420
2520   IF INKEY$=CHR$(27) THEN 2610
2530 NEXT CX
2540 '
2550 IF Y3>LY THEN Y1=LY:Y2=Y3 ELSE Y1=Y3:Y2=LY
2560 FOR CY=Y1 TO Y2 STEP R
2570   CX=X3 : GOSUB 1420
2580   IF INKEY$=CHR$(27) THEN 2610
2590   CX=LX : GOSUB 1420
2600 NEXT CY
2610 CX=X3 : CY=Y3
2620 PLAY "L64O4A" : GOSUB 1250 : GOSUB 1310
2630 RETURN
2640 '
2650 ' circle
2660 '
2670 R=SQR((CX-LX)^2+(CY-LY)^2)
2680 IF RF=1 AND MF$<>"B" THEN CIRCLE(LX,LY),R,FC
2690 RETURN
2700 '
2710 IF MF$="B" THEN K=120 ELSE K=60
2720 XX=CX:YY=CY
2730 C=CC : CC=1 : GOSUB 1250 : CC=C
2740 C=CL : CL=1 : GOSUB 1310 : CL=C
2750 FOR I=0 TO 2*PI STEP DG/(R/K)
2760   CX=LX+R*SIN(I) : CY=LY+R*COS(I)
2770   GOSUB 1420
2780   IF INKEY$=CHR$(27) THEN 2800
2790 NEXT I
2800 CX=XX : CY=YY
2810 PLAY "L64O4A" : GOSUB 1250 : GOSUB 1310
2820 RETURN
```

```
2830 '
2840 ' paint
2850 '
2860 PAINT (CX,CY),FC
2870 PLAY "L64O4A"
2880 RETURN
2890 '
2900 ' erase
2910 '
2920 C=CC : CC=1 : GOSUB 1250 : CC=C
2930 C=CL : CL=1 : GOSUB 1310 : CL=C
2940 '
2950 LINE (LX,LY)-(CX,CY),BC
2960 C=CC : CC=1 : GOSUB 1610 : CC=C
2970 IF INKEY$<>CHR$(27) THEN 2950
2980 '
2990 PLAY "L64O4A" : GOSUB 1250 : GOSUB 1310
3000 RETURN
3010 '
3020 ' box fill
3030 '
3040 IF CX=LX AND CY=LY THEN RETURN
3050 IF MF$="B" THEN 3090
3060 LINE (CX,CY)-(LX,LY),FC,BF
3070 RETURN
3080 '
3090 C=CC : CC=1 : GOSUB 1250 : CC=C
3100 C=CL : CL=1 : GOSUB 1310 : CL=C
3110 IF INKEY$=CHR$(27) THEN PLAY "L64O4A":GOSUB 1250:GOSUB 1310
:RETURN
3120   IF CX>LX THEN X1=LX:X2=CX ELSE X1=CX:X2=LX
3130   IF CY>LY THEN Y1=LY:Y2=CY ELSE Y1=CY:Y2=LY
3140   X3=INT(RND(1)*(X2-X1))+X1
3150   Y3=INT(RND(1)*(Y2-Y1))+Y1
3160   PSET (X3,Y3),FC
3170 GOTO 3110
3180 '
3190 ' circle fill
3200 '
3210 R=SQR((CX-LX)^2+(CY-LY)^2)
3220 IF MF$="B" THEN 3280
3230 FOR I=1 TO R
3240   CIRCLE (LX,LY),I,FC
3250 NEXT I
3260 RETURN
```

```
3270 '
3280 C=CC : CC=1 : GOSUB 1250 : CC=C
3290 C=CL : CL=1 : GOSUB 1310 : CL=C
3300 IF INKEY$=CHR$(27) THEN PLAY "L6404A":GOSUB 1250:GOSUB 1310
:RETURN
3310   RR=INT(RND(1)*(R+1))
3320   TT=RND(1)*2*PI
3330   X1=LX+RR*SIN(TT) : Y1=LY+RR*COS(TT)
3340   IF X1<0 OR X1>255 THEN RETURN
3350   IF Y1<0 OR Y1>191 THEN RETURN
3360   PSET (X1,Y1),FC
3370 GOTO 3300
3380 '
3390 ' color
3400 '
3410 SWAP FC,BC
3420 IF CC=1 THEN CC=CM ELSE CC=1
3430 PLAY "L6404A" : GOSUB 1250
3440 FOR T=1 TO 200 : NEXT T
3450 RETURN
3460 '
3470 ' Load & Save
3480 '
3490 SCREEN 0 : COLOR 15,4,4 : WIDTH 39 : KEY OFF
3500 '
3510 PRINT "*** Graphic Editor ***"
3520 PLAY "T128L1604A03A"
3530 PRINT
3540 INPUT "Load file name = ";F0$
3550 INPUT "Save file name = ";F1$
3560 PRINT
3570 PRINT "Are you sure ? (y/n)";
3580 I$=INPUT$(1)
3590 IF I$="N" OR I$="n" THEN 3530
3600 IF F0$="" THEN F0$=F1$ : GOTO 3710
3610 '
3620 CF=FC*16+BC
3630 BLOAD DV$+F0$ : GOSUB 3710
3640 FOR AD=0 TO 6143
3650   D=PEEK(AD+DA)
3660   VPOKE AD,D : VPOKE AD+8192,CF
3670 NEXT AD
3680 F0$=F1$
3690 RETURN
```

```
3700 '
3710 SCREEN 2,2,0 : COLOR FC,BC,BC : CLS
3720 RETURN
3730 '
3740 '  save
3750 '
3760 PLAY "L40AR8AR8A"
3770 SPRITE$(0)=""  : SPRITE$(1)="" : SPRITE$(2)=""
3780 '
3790 IF F0$="" THEN 3890
3800 FOR AD=0 TO 6143
3810   B=VPEEK(AD) : C=VPEEK(AD+8192)
3820   IF C=FC THEN B=255-B
3830   IF C=BC*16+FC THEN B=255-B
3840   POKE DA+AD,B
3850 NEXT AD
3860 '
3870 BSAVE DV$+F0$,DA,DA+6144
3880 '
3890 DEFUSR=&H3E  } ファンクションキーの定義を電源投入時と同様に戻す．知ってると便利です．
3900 DM=USR(DM)   }
3910 END
```

16K RAMへの対応

MSXは機種によってRAMの容量が異なります．もし，あなたのMSXが16K RAMの記憶容量を持つものだとすると，残念ながら，このグラフィックエディタのプログラムは動いてくれません．そこで16K RAMのMSX用にプログラムの変更をしなければなりません．

この変更によって，F7，F8のBox fillとCircle fillの命令，さらに画面データのロードとセーブが不可能になりますが，その他の機能はかわりなく動作します．

16K RAMへの変更リスト

```
120 CLEAR 300
140 '削除
170 '削除
190 COLOR FC,BC,BC : SCREEN 2,2,0
210 K1$=CHR$(223)+CHR$(191)+CHR$(127)+CHR$(222)
1040 IF I$=CHR$(127) THEN END
1180 ON I1 GOSUB 2080,2400,2670,2100
3010 行以降すべて削除
```

クイック リファレンス

コマンド

CIRCLE ……………………………… 16
CLOSE ……………………………… 21
COLOR ……………………………… 12
DRAW ……………………… 15, 74, 76, 118
LINE ……………………………… 15, 68, 70, 72
OPEN"GRP:" ……………………… 20
PAINT ……………………………… 16
PRESET……………………………… 15
PSET ……………………………… 15
PUT SPRITE ………… 78, 126, 143, 153
SCREEN ………………………… 11, 12
STICK ……………………………… 58
WIDTH ……………………………… 20

座標

n等分点の座標……………………… 36
回転移動による座標………………… 88
角度と距離による座標(極座標)‥44, 80, 114
画面上の座標における範囲………… 31
画面モード…………………………… 11
キャラクタ座標……………………… 40
高解像度グラフィックモード……… 31
座標の決め方………………………… 30
3次元座標から2次元座標への変換…… 90
相対座標……………………………… 51, 110
対称移動による座標の変化………… 40
中間点の座標………………………… 34
平行移動による座標の変化………… 37
マルチカラーモード………………… 33

描く

回転…………………………………… 88
拡大,縮小…………………………… 110
球……………………………………… 65
3次元曲線…………………………… 86
3次元グラフィックス ……………… 90, 164
周期関数による閉曲線〔SIN(T),COS(T)〕 …………… 84
スプライトパターン・ジェネレータ…… 171
直線の組み合わせ〔LINE〕 ………… 68
DATAの利用(一筆描き)〔READ/DATA, LINE〕………… 43
DATAの利用(複雑な形) ………… 70, 72
点滅…………………………………… 53
バイオリズム ………………………… 132

ブラシ効果 ……………………………………176
迷路……………………………………116, 137
文字の変形……………………………………94
物を消す………………………………………54
ランダム関数の利用〔RND(n)〕…………47

動かす

円周上の移動(時計の針)〔SIN(T), COS(T), TIME$〕……………114
回転(方位コンパス)〔DRAW〕…………122
キー入力による移動〔DIM(n)〕…………137
キー入力による移動〔STICK(n)〕 ………… 100, 120, 143
キー入力による移動〔INPUT$(l)〕 ……56
PUT SPRITE命令による移動 ………… 126, 143, 153
対称移動………………………………………39
追跡ルーチン ………………………………143
背景をスクロールさせる ………………124
平行移動………………………………………37
ボールの反射 ………………………………100
メータの表示 ………………………………122

並べる

円形〔SIN(T)/COS(T)〕……… 80, 88, 114
直線状……………………………………………51
マップで指定した形〔DIM(x,y)〕…143, 153
マス目状〔2重のFOR〜NEXT〕 ‥70, 74
ランダム〔RND(1)〕……………………………76
ランダム〔RND(1)*RND(1)〕 …………78

判定

POINT関数による判定 ……………106, 108
座標を使って位置を判定……………………60

その他

色と画面モード………………………………17
カラーコード…………………………………17
描きながら音を出す〔PLAY〕…………118
グラフィックキャラクタ………………… 21
グラフィックとキャラクタの混在……… 20
スプライト機能 ………………………22, 61
ハイスコア …………………………………104
表を作る ……………………………………104
補間 …………………………………………129
リカーシブコール ……………………97, 161

MESSAGE

書籍は常に「後書き」から読む，と決めて居る御人が可なりの割合で世間に存在する様子．斯くいう私もその一人であるからして，御同輩の気持ちはヨク分かる．だが本書は後書きを読んでも何の足しにも成りませんよ．まず第一に私は我が内に秘めたるネタを残滓無きまで絞り尽くして仕舞った．第二に此の文章はワープロで執筆した故か，自分も知らぬ様な漢字が多用されており，屹度，誤字脱字の類が有るに違い内．第三に，そしてこれが最大の理由ではありますが，このページ実は昨日まではこの世に無かった．とある人物のミスにより一夜のうちに惣然と出現してしまった場所なんです．大変ダァ，２ページも余っちまったヨゥ，との声に泰平の眠りを覚まされた私の立場も察して下さい．急に言われたって１時間やそこらでまともな「後書き」を捻り出すなんて私には無理無体というもの．御蔭でこんな文を読まざるを得なくなった読者の貴方，本当に御免なさい．これも皆あの××さんが悪いんです．

Writer：K. Sakurada

◆

読者の皆さん，ハーイ！　ボクでーす．ボクがこの本を書いた蓑島くんでーす．どーでしたか，ボクの作ったグラフィック・プログラム．なかなかキマッテいるでしょう．こういうセンスってのはねぇ，身につけようたってオイソレとは身につかないもんなんです．やっぱり生まれながらの天分が必要なんだよねぇ．ホラ，天才は天才を知るっていうでしょう，この本の良さがわかるキミはきっと才能豊かな人にちがいない．ボクはそう断言してハバカラないよ．

（遠い異国の空の下から届いた筆者の蓑島氏のメッセージに，多少の脚色を加えて再構成した文章であることを，ここにおことわりしておきます．編集Ｙ）

（ちょっとちょっとＹサン，ナニが「多少」の脚色ですか．これじゃボクがまったくバカみたいじゃないですか．ボクはもっと真面目な好青年なのに…….）

Writer：S. Minoshima

◆

ヒルベルト曲線との出会いが私の中に壁を作った．ドラゴンができた．ヒルベルトはドラゴンを２倍にして設定を変えたものなのさ．ドラゴンは入道雲のなかからやってくる黄金虫だとは思いませんか？

Co-Writer：M. Kanemitsu

◆

制作を担当いたしました．22才，独身，只今，赤丸上昇中のえりこです．私，トレスを特技として，身をたてていたのですが，ここだけの話ですけど，実は，曲線部に弱かったのです．ところが，その唯一の弱点をみごと克服し，あの，FM-7グラフィック・ワークブックでも，誰も描けなかったという，コサインカーブを，みごと描きあげてしまったのです．すっごいだろー．なんのこっちゃ!!

Creator:E.Tsutsui

◆

グラフィック・ワークブックをFM-7,MSXとたて続けに2冊，やっと終わった♡/ と思っていたのに…♡．この上原稿を書くはめになるなんて♡．いつも〝原稿だして〟とさけんでいる私ですが，文章を書くってたいへんですねぇ～♡．(これで5行…うまった) 最後に一言，ステキな方からのアンケート葉書きまってます♡．(写真を併えて)

Creator:S.Fujita

◆

皆さん御苦労様でした．

協力:OKURI

◆

なっなんだ/ たっ単行本にへっ編集後記があるなんて/ とおどろかれた方もいるでしょうね．

このページは，よりユーザーフレンドリーな本づくりを目刺すアスキーからのメッセージなのです．けっして，ぜったいに穴うめなどではありません．

えっまだ行数あるの/

えーっと，ずいぶん，最近，マイコンもメジャーになってきました．コンピュータ・グラフィック，ゲームから，統計処理，人工知能もどきまでいろんなソフトウェアが存在しています．そんなソフトウェアを買ってくれれば，それなりに用は足りてしまいます．しかし，グラフィックなりBASICなりを勉強してみようかって人のためにこれらを楽しみながら身に付けていただきたく本書は作られました．どうでしたか？

どうか，プログラミングの楽しさを知ったからといって，いつもいつも深夜一人でねちねちとプログラムを作っては，一人で楽しむなんてプログラマーにはならないでほしいのです．

なお，本書への希望，意見などアンケートハガキに書いてお送りくだされば幸いです．

Super ME-HER Editor:Suis-je

参考文献

〔1〕「PC-8801　グラフィックスのすべて」
工藤丈彦　他
アスキー出版局

〔2〕「PC-9801　ガイドブック」
榊 正憲　他
アスキー出版局

〔3〕「コンピュータ　グラフィックス」
DAVID F.ROGERS
J.ALAN ADAMS
山口富士夫　訳
日刊工業新聞社

〔4〕Oh！FM別冊 「FILES Oh！FM」
日本ソフトバンク

〔5〕「バイオリズムとはなにか」
田多井吉之介
講談社ブルーバックス

〔6〕「MSXビギナーズBASIC」
児玉真之
アスキー出版局

MSX グラフィック・ワークブック

1984年5月5日　初版発行
1985年10月25日　第1版第2刷発行
定価1,500円

著　者　桜田幸嗣・蓑島聡
発行者　塚本慶一郎
発行所　株式会社アスキー
〒107 東京都港区南青山5－11－5住友南青山ビル5F
振　替　東京4-161144
電　話　03－486－7111(代表)

編集担当　竹山正寿
表紙担当　郷 啓子
印刷　株式会社加藤文明社印刷所

ISBN4-87148-730-X C0055 ¥1500E

デザイン・シート

x 座標

y 座標	0	20	40	60	80	100	120	140	160	180	200	220	240
0													
20													
40													
80													
100													
120													
140													
160													
180													

キャラクタコード表

コード 10進	*キャラクタ	対応するキー	コード 16進	コード 10進	キャラクタ	コード 16進	コード 10進	キャラクタ	コード 16進	コード 10進	キャラクタ	コード 16進	コード 10進	キャラクタ	コード 16進	コード 10進	キャラクタ	コード 16進	コード 10進	キャラクタ	コード 16進	コード 10進	キャラクタ	コード 16進
0	ヌル	—	&H00	32		&H20	64	@	&H40	96	`	&H60	128	♠	&H80	160		&HA0	192	タ	&HC0	224	た	&HE0
1	月	CTRL A	&H01	33	!	&H21	65	A	&H41	97	a	&H61	129	♥	&H81	161	。	&HA1	193	チ	&HC1	225	ち	&HE1
2	火	CTRL B	&H02	34	″	&H22	66	B	&H42	98	b	&H62	130	♣	&H82	162	「	&HA2	194	ツ	&HC2	226	つ	&HE2
3	水	CTRL C	&H03	35	#	&H23	67	C	&H43	99	c	&H63	131	♦	&H83	163	」	&HA3	195	テ	&HC3	227	て	&HE3
4	木	CTRL D	&H04	36	$	&H24	68	D	&H44	100	d	&H64	132	○	&H84	164	、	&HA4	196	ト	&HC4	228	と	&HE4
5	金	CTRL E	&H05	37	%	&H25	69	E	&H45	101	e	&H65	133	●	&H85	165	・	&HA5	197	ナ	&HC5	229	な	&HE5
6	土	CTRL F	&H06	38	&	&H26	70	F	&H46	102	f	&H66	134	を	&H86	166	ヲ	&HA6	198	ニ	&HC6	230	に	&HE6
7	日	CTRL G	&H07	39	'	&H27	71	G	&H47	103	g	&H67	135	ぁ	&H87	167	ァ	&HA7	199	ヌ	&HC7	231	ぬ	&HE7
8	年	CTRL H	&H08	40	(	&H28	72	H	&H48	104	h	&H68	136	ぃ	&H88	168	ィ	&HA8	200	ネ	&HC8	232	ね	&HE8
9	円	CTRL I	&H09	41	)	&H29	73	I	&H49	105	i	&H69	137	ぅ	&H89	169	ゥ	&HA9	201	ノ	&HC9	233	の	&HE9
10	時	CTRL J	&H0A	42	*	&H2A	74	J	&H4A	106	j	&H6A	138	ぇ	&H8A	170	ェ	&HAA	202	ハ	&HCA	234	は	&HEA
11	分	CTRL K	&H0B	43	+	&H2B	75	K	&H4B	107	k	&H6B	139	ぉ	&H8B	171	ォ	&HAB	203	ヒ	&HCB	235	ひ	&HEB
12	秒	CTRL L	&H0C	44	,	&H2C	76	L	&H4C	108	l	&H6C	140	ゃ	&H8C	172	ャ	&HAC	204	フ	&HCC	236	ふ	&HEC
13	百	CTRL M	&H0D	45	−	&H2D	77	M	&H4D	109	m	&H6D	141	ゅ	&H8D	173	ュ	&HAD	205	ヘ	&HCD	237	へ	&HED
14	千	CTRL N	&H0E	46	.	&H2E	78	N	&H4E	110	n	&H6E	142	ょ	&H8E	174	ョ	&HAE	206	ホ	&HCE	238	ほ	&HEE
15	万	CTRL O	&H0F	47	/	&H2F	79	O	&H4F	111	o	&H6F	143	っ	&H8F	175	ッ	&HAF	207	マ	&HCF	239	ま	&HEF
16	π	CTRL P	&H10	48	0	&H30	80	P	&H50	112	p	&H70	144		&H90	176	ー	&HB0	208	ミ	&HD0	240	み	&HF0
17	┴	CTRL Q	&H11	49	1	&H31	81	Q	&H51	113	q	&H71	145	あ	&H91	177	ア	&HB1	209	ム	&HD1	241	む	&HF1
18	┬	CTRL R	&H12	50	2	&H32	82	R	&H52	114	r	&H72	146	い	&H92	178	イ	&HB2	210	メ	&HD2	242	め	&HF2
19	┤	CTRL S	&H13	51	3	&H33	83	S	&H53	115	s	&H73	147	う	&H93	179	ウ	&HB3	211	モ	&HD3	243	も	&HF3
20	├	CTRL T	&H14	52	4	&H34	84	T	&H54	116	t	&H74	148	え	&H94	180	エ	&HB4	212	ヤ	&HD4	244	や	&HF4
21	┼	CTRL U	&H15	53	5	&H35	85	U	&H55	117	u	&H75	149	お	&H95	181	オ	&HB5	213	ユ	&HD5	245	ゆ	&HF5
22	│	CTRL V	&H16	54	6	&H36	86	V	&H56	118	v	&H76	150	か	&H96	182	カ	&HB6	214	ヨ	&HD6	246	よ	&HF6
23	─	CTRL W	&H17	55	7	&H37	87	W	&H57	119	w	&H77	151	き	&H97	183	キ	&HB7	215	ラ	&HD7	247	ら	&HF7
24	┌	CTRL X	&H18	56	8	&H38	88	X	&H58	120	x	&H78	152	く	&H98	184	ク	&HB8	216	リ	&HD8	248	り	&HF8
25	┐	CTRL Y	&H19	57	9	&H39	89	Y	&H59	121	y	&H79	153	け	&H99	185	ケ	&HB9	217	ル	&HD9	249	る	&HF9
26	└	CTRL Z	&H1A	58	:	&H3A	90	Z	&H5A	122	z	&H7A	154	こ	&H9A	186	コ	&HBA	218	レ	&HDA	250	れ	&HFA
27	┘		&H1B	59	;	&H3B	91	[	&H5B	123	{	&H7B	155	さ	&H9B	187	サ	&HBB	219	ロ	&HDB	251	ろ	&HFB
27	╳		&H1C	60	<	&H3C	92	¥	&H5C	124	|	&H7C	156	し	&H9C	188	シ	&HBC	220	ワ	&HDC	252	わ	&HFC
29	大		&H1D	61	=	&H3D	93	]	&H5D	125	}	&H7D	157	す	&H9D	189	ス	&HBD	221	ン	&HDD	253	ん	&HFD
30	中		&H1E	62	>	&H3E	94	^	&H5E	126	~	&H7E	158	せ	&H9E	190	セ	&HBE	222	゛	&HDE	254		&HFE
31	小		&H1F	63	?	&H3F	95	_	&H5F	127		&H7F	159	そ	&H9F	191	ソ	&HBF	223	゜	&HDF	255		&HFF

(コード 1～31: コントロール・キャラクタ)

* このグラフィックキャラクタをPRINT，LPRINT命令で表示する時には，CHR$（1）+CHR$（n）とする．

ASCII
アスキーブックス

ISBN4-87148-730-X C0055 ¥1500E 定価1,500円